REVISTA PARA USUARIOS DE

Drean Commodore

AÑO 1 Nº 6 ₳ 2.00 REP. ARGENTINA



Ventajas del Drive 1541

Programas
PROTECTOR DE ARCHIVOS
PRODE MUNDIAL

Algoritmos
COMO CLASIFICAR DATOS

Subrutinas
SISTEMA OPERATIVO

16, 64 y 128
ALMACENAMIENTO DE SOFT

¿El mejor lenguaje?

MAYO 1986 Nº 14 ₳ 2.30 REP. ARGENTINA

K64

COMPUTACION PARA TODOS



Argentina Siglo 21

15 Programas Inéditos

Interfase Kempston Para Spectrum

Suplemento Educativo Para C64, TI99, CZ, TK y MSX

CPM en la C-128

Cómo Ganar Un Lingote de Oro

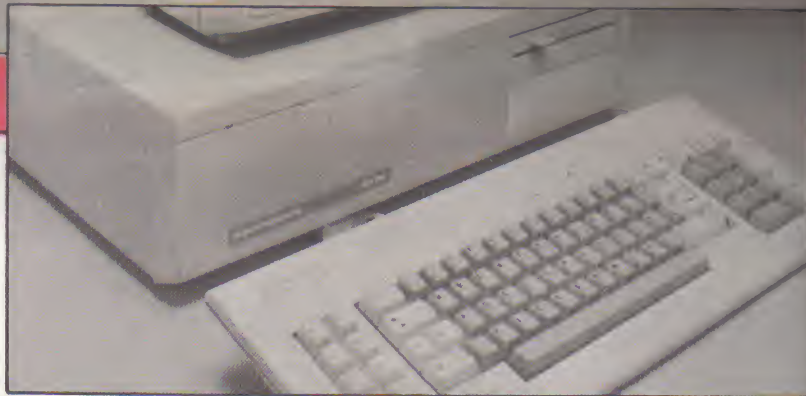
SUMARIO

NOTAS TECNICAS

PROGRAMAS

REVISION DE SOFT

SECCIONES FIJAS

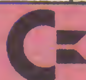


Comentamos las características más sobresalientes de la nueva máquina.



Este periférico es ideal para trabajar con archivos relativos. Dispone de mayor densidad para el almacenamiento de datos que el Dattaset.

Nos adelantamos a las olimpíadas de 1986 comentándoles Summer Games, además de otros tres divertidísimos juegos.

Drean  commodore AÑO 1 Nº 6 MAYO DE 1986

Director General
Ernesto del Castillo

Director Editorial
Cristián Pusso

Director Periodístico
Fernando Flores

Director Financiero
Javier Campos Malbran

Arte y Diagramación
Fernando Amengual
Tamara Migelson

Coordinador
Ariel Testori

Redacción
Cristián Parodi

Departamento de Avisos
Oscar Devoto

Departamento de Publicidad
Guillermo Gonzalez Aldalur

Drean Commodore es una Revista mensual editada por editorial PROEDI S.A. [illegible] 720, 5º Pis. (1017) Buenos Aires. Tel. 46-2886 y 49-7130. Rep. [illegible] Intelectual E.T., M. Registrada.

Precio de este ejemplar. A 2.
Impresion: Calcotam. Fotocromo tapa: Columbia. [illegible]
Los ejemplares atrasados se venderan al precio del ultimo [illegible]

Distribuidor en Capital: [illegible] DGP [illegible] Capital Federal. [illegible]

LENGUAJES Y MUCHO MAS

Continuamos describiendo los contenidos de las distintas notas técnicas que acompañan a este número.



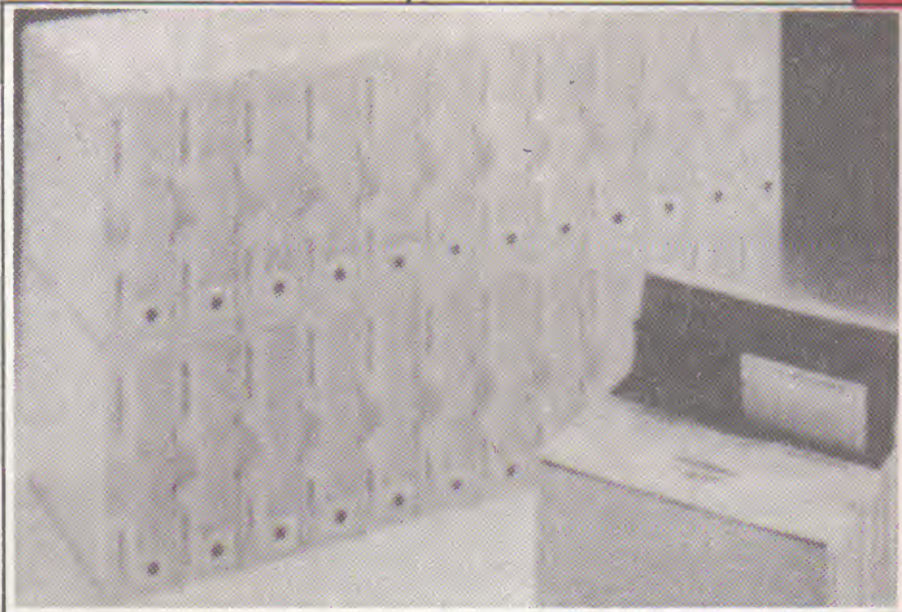
Bien, el numero seis llegó y junto con él seguimos entregando más notas y más programas utilitarios.
En honor al nombre de esta sección debemos decir, ¿que es un programa utilitario?
Sin entrar en el "verso cruel", un programa utilitario es aquel que nos ayuda a realizar una determinada tarea.
Por ejemplo, PROTECTOR DE ARCHIVOS es un programa que nos permite proteger el disco para que, no podamos borrar programas almacenados en él o que nadie los almacene sobre el.
En la nota referente al drive de la C-64 realizamos una descripción sobre este fiable periférico.
Hablamos un poco de su buffer, quien permite compatibilizar velocidades de transferencia de datos, entre éste y la computadora. En otras palabras, demora la información para que ninguno de los dos se olvide algo.
Relatamos un poco sobre su controlador de disco, el cual se encarga de manejar la unidad para que opere correctamente.
Marcamos las diferencias de trabajar con archivos secuenciales y directos. Estas dos palabras se refieren al acceso de la información en el disco.
En el primero (secuencial) para acceder a un determinado valor debemos pasar, obligatoriamente, por todos los anteriores.
En cambio, en el segundo (directo) al valor se accede directamente, sin necesidad de pasar por los anteriores.
Tocamos un poco el tema de programar la unidad para que esta opere más rápido.
Seguimos con otro tema; clasificación u ordenamiento de datos.
Aquí explicamos distintos algoritmos para que puedan ordenar la información (por ejemplo de mayor a menor) usando la computadora como herramienta.
¿Han dicho algrotitmo o algo por el estilo? No, es algoritmo. Se refiere a un método para resolver un problema. No hace a ningún lenguaje en particular. Es universal. Sólo se encarga de describir los pasos necesarios para solucionar un determinado problema.
Y mencionamos lenguajes. Claro, no nos referimos al inglés o al francés o al jerigonzo.
Al igual que éstos, se utilizan para comunicarse. Los que aquí describimos son usados para relacionarse con la computadora.
¿Quieren saber cuál es el mejor? Para eso lean la nota.Seguimos con las investigaciones. Nos preguntamos cómo se almacenan los programas en memoria. Lo descubrimos y les transmitimos los resultados.
Allí verán que hay ciertos códigos, cómo se relaciona la información, dónde se almacena, y más cosas interesantes.
Lo bueno de todo es que es universal, es válido para la C-16, C-64 y C-128.
Seguimos con mapa de memoria y con las subrutinas de la Drean Commodore 64.
¿Qué son las subrutinas? En más de una ocasión, cuando programamos, debemos ejecutar un procedimiento varias veces.
En lugar de escribirlo todas las veces que lo necesitamos, lo escribimos una vez y accedemos a él cuando lo requerimos.
Por supuesto continuamos entregándoles nuevos trucos para aplicarlos, noticias frescas y nuestra revisión de software.

Provoice-sintetizador de voz para la C-64

Provoice es un nuevo sintetizador de voz recientemente lanzado para la C-64. Puede "conversar" en idioma inglés. Prácticamente su vocabulario es ilimitado. El dispositivo está formado por una memoria ROM donde se encuentra el programa y los fonemas y un pequeño amplificador de audio. Se conecta a la Port del usuario. Además suministra en total 13 nuevos comandos incluyendo HELP para tener una rápida referencia de éstos. Por ahora sólo está disponible en USA.

Sintetizador de voz

La empresa Votrax ha lanzado su nuevo sintetizador de voz. Este puede hablar fluidamente, pronunciar cada palabra y números y símbolos.
El dispositivo se encuentra en una pequeña caja de 4x5 pulgadas, la cual se conecta al port del usuario.
Dispone de comandos que permiten amplificar la voz, conversar, almacenar en disco, etc.
El único inconveniente que posee es su precio. La unidad más toda la documentación requerida cuesta u$s 100



Almacenamiento

No sólo se desarrollan nuevos productos de soft y hard, también se desarrollan nuevos módulos que permiten un mejor almacenamiento de los soportes magnéticos.
Por ejemplo, una nueva caja para almacenar diskettes de 5-1/4", que consiste en una serie de cajas interconectadas.
Posee un mecanismo que levanta los discos para que puedan ser tomados cómodamente.
Por ahora sólo se encuentra disponible en Europa.



Fligth Simulator II

SubLogic, creadora del famoso simulador de vuelo, ha lanzado seis nuevos escenarios para este juego y el extraordinario "Jet".
De esta manera la capacidad de ambos simuladores aumenta notablemente, aunque de por sí ya es más que suficiente.

Tanto Jet como Flight Simulator II pueden ser adquiridos en nuestro país al igual que el disco que contiene los nuevos paisajes.

C-64 DMS

Recientemente lanzado en Estados Unidos, DMS (Disk Management System) es un utilitario que permite administrar los recursos del drive 1541.
Básicamente está formado por tres programas para realizar distintas operaciones.
Ellos permiten listar el directorio del disco sobre la pantalla mientras ejecuta un comando de disco.
Permite, además, proteger los datos del disco a través de claves. De esta manera el acceso a la información es solamente realizado por el usuario, ingresando la clave correcta.
El último programa permite un fácil manejo y mantenimiento de nuestras bases de datos.
A través de un menú, el usuario puede acceder a cada uno de estos programas y, también, enviar comandos al disco en el momento que lo desee.

COMMODORE PC-10



Comentamos las características más sobresalientes de la nueva máquina. Su sistema operativo es el MS DOS, que le permite ser compatible con los programas de IBM PC.

Una de las noticias que hemos publicado en este número se refiere a la decisión, por parte de la empresa Drean, de fabricar la nueva PC de Commodore: su PC-10 y PC-20.
Primero fueron la VIC 20 y la C-64.
Luego se desarrolló la C-16, en donde se visualizaba la mejora de éste con respecto a los anteriores. Entre otras cosas se puede observar su nuevo Basic V3.5.
No transcurrió demasiado tiempo desde este lanzamiento, cuando Commodore anunció su nueva C-128 juntamente con la nueva disketera 1571, su impresora 1000 y el monitor 1902.
De esta manera se iba reduciendo el espacio que alejaba a la empresa de sus objetivos inmediatos: la Commodore Amiga y la PC.
Con el lanzamiento de la Amiga se observó una diferencia en tecnología que nos hacía suponer que el progreso aumentaba de equipo en equipo en forma exponencial.
Una de las características de éste es la capacidad de generar 4096 colores distintos y, además, poder generar voz humana.
Con respecto al Hardware del sistema, Commodore colocó un microprocesador de 16 bits (como lo es el 68000) aumentando la velocidad de ejecución.
Hace poco tiempo se produjo un nuevo lanzamiento: la PC.
Primeramente mencionaremos el diseño modular del que fue "víctima" la PC.
Cada uno de ellos se encuentra dentro de la CPU, la cual contiene la unidad central de procesamiento.
Aquí se encuentran los procesadores, la memoria principal, los dos drives destinados para almacenamiento masivo de la información y los distintos "slots" para la expansión del equipo.
Gracias a ellos es posible agregar modems, controladores para disco duro, mouse, teclado gráfico, expansión de memoria, etc.

A través de una interface RS232 que viene incluida dentro del equipo, se pueden conectar impresoras paralelo tipo centronics, plotters, conectar otra computadora (a través del modem)

En lo que respecta al sistema operativo de la PC, éste es el MS DOS. De esta manera la PC permite ser compatible con todos los programas escritos para la PC de IBM.
Base de datos, planillas electrónicas, lenguajes, procesadores de texto, son algunos de ellos.
La PC-10 utiliza el microprocesador 8088

Especificaciones técnicas:

CPU:	Microprocesador 8088. Co-procesador 8087 (opcional).
CLOCK:	4.77 MHz
MEMORIA:	RAM principal de 256 Kb, internamente expandible a 512 Kb, externamente expandible a 640 Kb utilizando todos los slots. ROM de 8 Kb.
MONITOR:	12" alta resolución (fósforo verde) 14" media resolución. Tamaño de cada caracter: 9x14 - 80 columnas por 25 líneas.
ALMACENAMIENTO:	Dos drives para discos flexibles de 360 Kb (se puede conectar un controlador de disco duro de 10 Mb. Formato compatible con IBM (40 tracks, 9 sectores).
ALMACENAMIENTO EXTERNO:	El controlador interno de los drives puede manejar hasta cuatro unidades. Permite, también, trabajar con dos discos duros tipo Winchester de 70 Mb.
TECLADO:	Teclado separable. 85 teclas incluídas 10 teclas de función.
INTERFACE:	Port para impresoras paralelo (centronics)
SISTEMA	RS232 C
OPERATIVO:	Monitor monocromático.
	MS-DOS
LENGUAJE:	GW BASIC Opcionales: Cobol, Basic, Pascal, Macro Assembler, etc.

NUEVOS DESARROLLOS



de 16 bits juntamente con una capacidad de 256 Kb de memoria libre. Esta puede; ser fácilmente expandida a 512 Kb (internamente) o a 640 Kb (utilizando los slots).

La capacidad de almacenamiento del disco es de 360 Kb (discos flexibles de 5 1/4").

La PC-20 tiene las mismas características para el almacenamiento en disco flexible y, además, suministra un controlador para disco duro de una capacidad de 10 Mb.

LENGUAJES DE PROGRAMACION

No sólo en Basic se puede programar la C-64. Por eso, describimos los distintos lenguajes en que puede trabajar la Drean Commodore.

Como todos saben cada vez que activamos la computadora, el intérprete Basic que se encuentra dentro de ella está listo para "correr". Lo que no todos saben es que el Basic, es uno de los tantos lenguajes que pueden utilizarse para desarrollar nuestros programas.

Logo, Pascal, C, Forth, Comal, Promal son algunos de los lenguajes de programación que pueden "correrse" en la C-64. Cada uno de ellos fue desarrollado teniendo en cuenta distintos objetivos.

Por ejemplo Pascal, es un lenguaje orientado hacia la programación estructurada, Logo fue desarrollado con fines educativos, Basic para los que se inician, y así cada lenguaje con su objetivo.

¿De qué dependerá, pues, la elección del lenguaje y cuál será el mejor?

Empecemos contestando la segunda pregunta y viendo si realmente existe el lenguaje "superior". De hecho buscar el mejor significa elegir de entre muchos el que más se adecúe a los requisitos exigidos.

Por supuesto que todos ellos deben pertenecer al mismo conjunto, es decir deben poder ser comparables. Pero como cada lenguaje fue desarrollado teniendo en cuenta distintos objetivos, creemos que no es posible compararlos y, por lo tanto, tampoco es posible elegir el mejor.

Al mismo tiempo estamos contestando la primera pregunta. Evidentemente, la elección del lenguaje dependerá de la aplicación que le daremos.

La mayoría de las personas (por no decir todas) que se inician en la computación lo hacen con el Basic. Después de todo, es el lenguaje que acompaña a todas las computadoras hogareñas.

En el caso de la C-64, éste es el Basic V2.0 desarrollado por la empresa Microsoft, allá por el año 1977. Lamentablemente no tiene comandos orientados al manejo de gráficos y sonidos. Estos fueron algunos de los motivos por los que se desarrollaron Basic mejorados, como el caso del Simon's Basic, descripto en anteriores números.

A continuación describiremos las características más importantes de los lenguajes de programación disponibles para la C-64. Antes queremos decir que, a la hora de ejecutarse, siempre se traducen en código máquina para que la computadora lo pueda comprender.



Logo

Es uno de los más populares lenguajes con fines educativos. Fue desarrollado por un grupo de computadores científicos, encabezado por Seymour Papert, en el año 1960. Papert y su grupo buscaban un lenguaje que fuera fácil de usar por los niños.

Una parte del lenguaje contiene comandos gráficos, desarrollados a través del movimiento de una tortuga (el primer uso del lenguaje utilizaba un robot similar a una tortuga, de ahí el nombre).

Todo el lenguaje está basado en LISP (LISt Processing) lenguaje de alto nivel utilizado para manipulación de texto y análisis y, además, utilizado en la investigación de inteligencia artificial.

Logo contiene comandos denominados "primitivos". Estos son utilizados para realizar un determinado programa.

Sin embargo los "primitivos" pueden combinarse para crear nuevos comandos. De esta manera se logra extender el lenguaje.

Otra de las características de este lenguaje es su estructura modular, el cual le permite al programador definir "bloques" de programas formando, así, toda una estructura.

El Logo desarrollado para Commodore es sumamente potente. Posee comandos para generar sonidos y gráficos, incluyendo los sprites. Actualmente es posible encontrarlo en nuestro mercado.

Pascal

Este es otro de los lenguajes que impactó en la educación. En este caso a nivel universitario. Su nombre fue puesto en honor del matemático francés Blaise Pascal. El lenguaje fue creado en 1970.

Orientado hacia la programación estructurada, Pascal también permite una estructura modular de los programas, respetando que cada módulo tiene una sola entrada y una sola salida.

Una de las características más sobresalientes del lenguaje es la recursividad e iteración.

Hay, por lo menos, tres versiones de Pascal para la C-64. Uno de ellas es Super Pascal desarrollado por Abacus.

Pilot

Pilot fue desarrollado en la Universidad de California por John Starkweather en el año 1968. Es la sigla de Programmer Inquiry Learning or Teaching (preguntas programadas aprendiendo o enseñando). Su objetivo apuntaba a utilizar la computadora como una herramienta más para la educación.
El lenguaje es fácil de utilizar. Permite crear buenos gráficos y, además, dispone de un fácil manejo para tomas de decisión en bifurcaciones a través de true (verdad) y false (falso).
Existe una versión de Pilot para la C-64.

Forth

Al igual que Pascal, existen varias versiones para la C-64.
Desarrollado en 1960 por Charles Moore, Forth apunta a la programación estructurada. Como en Logo, se pueden combinar las instrucciones para crear nuevos comandos.
Al principio fue utilizado para controlar telescopios. Durante el transcurso del tiempo, Forth ha sido perfeccionado, llegándose a la versión actual.
Una característica importante de este lenguaje es que tiene un Stack en donde se realizan las operaciones aritméticas. La notación que utiliza es la polaca inversa. Es decir primero se ingresan los operandos y luego la operación.

Comal

Fue desarrollado por Borge Christensen en el año 1970. Actualmente es el lenguaje de computación oficial en las escuelas de Dinamarca e Irlanda.
Es fácil de aprender siendo su estructura muy similar al Logo. Para la C-64 existen dos versiones: Comal.014 y Comal 2.0.

C

Creado en 1972 por Dennis Ritchie, el lenguaje C ha incrementado su popularidad, especialmente en los últimos años. Es sumamente potente y puede ser transportado de una computadora hacia otra.
Los programas escritos en C se ejecutan mucho más rápido que los desarrollados en otros lenguajes (excepto aquellos escritos en lenguaje máquina). Una vez desarrollado el programa, éste puede ser fácilmente trasladado a otras máquinas para ser ahí ejecutado.

Promal

Creado por Systems Management Associates, Promal son las siglas de Programmer's Micro Application Lenguage. Es un lenguaje compilable siendo su estructura muy similar a la de Pascal o C.
En él se distinguen tres secciones en donde se definen variables, procedimientos y funciones y, por último, el módulo principal o programa Promal.
El lenguaje viene acompañado por un sistema operativo (denominado ejecutivo) cuya función es administrar los recursos del computador.
Además de los lenguajes que ya mencionamos, existen otros como el Cobol (Common Business Oriented Language) y Fortran (Formula Translation). Ambos lenguajes fueron desarrollados en el año 1950 y cada uno tiene distintas aplicaciones. El primero fue diseñado para aplicaciones comerciales mientras que el segundo fue desarrollado con fines científicos.

PROTECTOR DE ARCHIVOS

Con este software podremos proteger los programas y todo el disco, además de realizar otras funciones de la C-64 y la C-16.

PROTECTOR DE ARCHIVOS

1) DIRECTORIO
2) PROTEGE TODOS LOS PROGRAMAS
3) PROTEGE UN PROGRAMA
4) DESPROTEGE UN PROGRAMA
5) BORRA UN PROGRAMA
6) DESPROTEGE EL DISCO
7) BORRA UN PROGRAMA
8) FIN

SELECCIONE SU OPERACION

¿Cuantas veces hemos borrado por error programas o archivos que no debíamos borrar? Por supuesto que, lo hecho, hecho está.
Siempre nos hacemos la promesa de que la próxima vez prestaremos mayor atencion. Sin embargo...
Para que nunca más nos suceda esto les presentamos un utilitario que permite proteger archivos o programas determinados, proteger todos los programas o archivos del disco, desproteger los programas o archivos seleccionados, proteger el disco contra la escritura, sin necesidad de poner la etiqueta de protección y, finalmente, desprotegerlo.
Cada programa que protejamos no podrá ser borrado del directorio del disco y, en caso de proteger todo el disco, no podremos almacenar ningún programa sobre él.
Una vez que hayan cargado y ejecutado el programa, se imprimirá el menú principal, que está formado por:
1) Directorio
2) Protege todos los programas
3) Protege un programa
4) Desprotege un programa
5) Protege todo el disco
6) Desprotege todo el disco
7) Borra un programa
8) Borra un programa
8) Fin

Debemos oprimir el número de la opción seleccionada.
La primera imprime en pantalla el directorio del disco. En este bloque se le informa al usuario cómo puede suspender temporalmente el listado del directorio (oprimiendo la tecla SHIFT).
La segunda opción protege todos los programas (programas y archivos) del disco. A medida que se va ejecutando, se imprime el track y sector que se están protegiendo.
El tercer modo protege un programa determinado. Este bloque también le informa al usuario cómo puede hacer para salir de este modo o para imprimir el directorio del disco (esto se logra a través de F1 y de F7).
En caso de que el programa no se encuentre en el disco, se imprimirá el correspondiente mensaje. Si, en cambio, el programa está en el disco, pero ya ha sido protegido, nos informaremos con "EL PROGRAMA YA ESTABA PROTEGIDO".
Si todo es correcto, se protegerá el programa deseado.
A través del cuarto modo podemos desproteger los programas que anteriormente protegimos. También se informa cómo se puede imprimir el directorio, volviendo luego a este modo, y como se retorna al menú principal.
En caso de que el programa a desproteger no se halle en el disco, ocurrirá lo mismo que en el bloque anterior (se imprime el mensaje correspondiente).
Si el programa se encontraba desprotegido, se imprimirá "YA ESTABA DESPROTEGIDO". En caso de no ocurrir estos dos, se procederá a desproteger el programa.
La quinta opción protege todo el disco. A través de F1 podemos regresar al menú principal. Una vez que el disco esté protegido, no podremos almacenar programas sobre él.
La sexta opción realiza el procedimiento inverso al anterior; es decir desprotege el disco.
La séptima opción permite borrar programas que no estén protegidos. Caso contrario se imprimirá el correspondiente mensaje.
Finalmente, la octava regresa al Basic
Los programas protegidos se distinguen de los demás con un simbolo "<" al

final del nombre de él.

COMO TIPEAR EL PROGRAMA

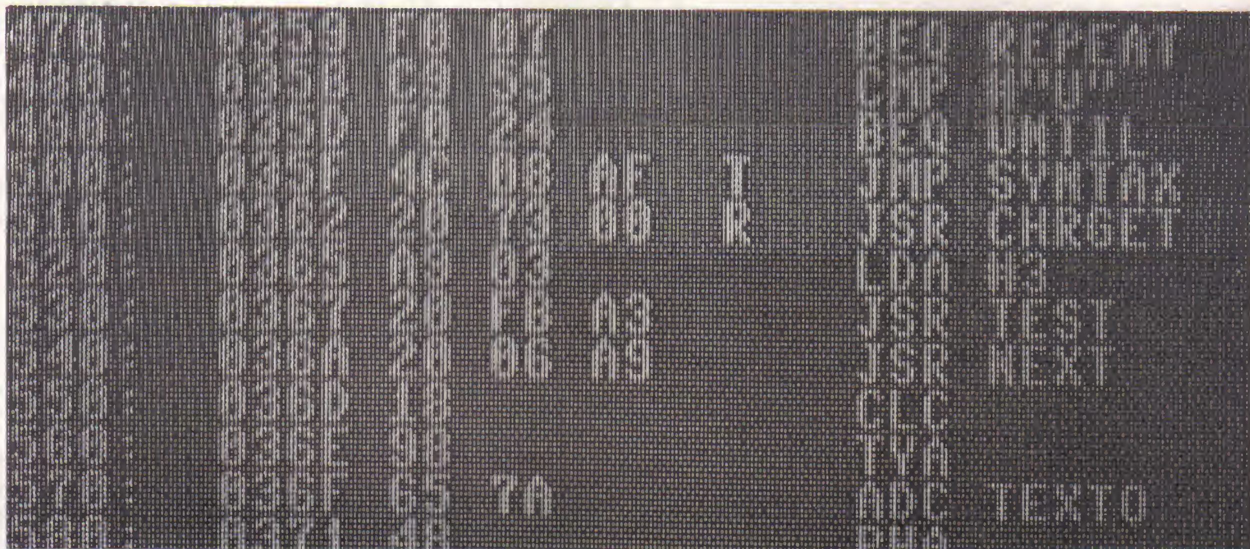
Antes de describir como se debe ingresar el programa, debemos decir que es lo que tienen que hacer los usuarios de la C-16 para poder utilizarlo. Ellos, sólo ellos, deben cambiar la línea 20 por:
20 FORA = 1TO8: KEYA,"
":NEXT:KB=239:SF=1347:KL=
198.F1=4:F7=3
Pasemos pues a describir el programa. Como habrán notado, los caracteres especiales no figuran en el listado. En lugar de ellos se encuentran las teclas que se deben oprimir para general esos caracteres. Ellos son:
cr ab: Se debe oprimir la tecla que mueve el cursor hacia abajo.
cr iz: Se debe oprimir la tecla que mueve el cursor hacia la izquierda.
es: Se debe oprimir la tecla espaciadora
clr: Se debe oprimir SHIFT y la tecla de CLR/HOME simultáneamente.
blk: Se debe oprimir CTRL y 1 simultáneamente.
red: Se debe oprimir CTRL y 3 simultáneamente.
rvs on: Se debe oprimir CTRL y 9 simultáneamente.
NO se deben teclear los signos de admiración que rodean a estos caracteres. Por ejemplo, ¡40 es! significa que se debe oprimir cuarenta veces la tecla espaciadora, ¡3 cr iz! significa que se debe oprimir tres veces la tecla que mueve el cursor hacia la izquierda, etc. Como en otra oportunidad mencionamos, de esta forma evitamos confusión sobre los caracteres especiales.
El listado principal es el que sigue a continuación.

```
1   REM PROTECTOR DE ARCHIVOS
5   PRINT"!clr!":Q$=CHR$(13)
10 U$="!rvs on!!40 es!":KB=198:SF=653:KL=203:NS=12
20 F1=4:F7=3:POKE53281,12:POKE53280,6:REM CUIDADO C-16 !!!
50 GOSUB1240:GOTO1080
60 PRINT#15,"U1";2;0;T;S:GOTO670
70 PRINT#15,"B-P 2 0":PRINT#15,"U2";2;0;T;S:GOTO670
80 T=PEEK(681):S=PEEK(682):RETURN
90 REM *** PROT.TODOS LOS PROGRAMAS ***
100 PRINT"!clr!"SPC(NS-4)"PROTEGE TODOS LOS PROGRAMAS":PRINTU$:
    GOSUB660:T=18:S=1
110 P=2:GOSUB60
120 GET#2,A$:T1=ASC(A$+CHR$(0)):GET#2,A$:S1=ASC(A$+CHR$(0))
130 FORI=0TO7:PRINT#15,"B-P";2;P+32*I
140 GET#2,A$:A=ASC(A$+CHR$(0)):IFA=.THEN170
150 IFAAND64THEN170
160 PRINT#15,"B-P";2;P+32*I:PRINT#2,CHR$(AOR64);
170 NEXT:GOSUB70:IFETHENRETURN
180 PRINT"TRACK"T"SECTOR"S" ESTA PROTEGIDO":T=T1:
    S=S1:IFTTHEN110
190 GOSUB730:GOSUB740:RETURN
200 REM *** DIRECTORIO ***
210 H$=CHR$(18):PRINT"!clr!!rvs on!OPRIMA [SHIFT] PARA
    SUSPENDER EL LISTADOQ
220 GOSUB660:PRINT#15,"M-R"CHR$(144)CHR$(7)CHR$(23)
230 FORI=0TO22:GET#15,A$:H$=H$+(A$+CHR$(0)):NEXT:
    PRINTH$"!blk!":POKE140,1:T=18:S=1
240 GOSUB60:SYS828:GOSUB80:IFPEEK(SF)THENWAITSF,1,1
250 IFTTHEN240
260 PRINT#15,"B-R"CHR$(150)CHR$(2):GET#15,LO$:
    PRINT#15,"M-R"CHR$(252)CHR$(2)
270 GET#15,HI$:
    PRINTASC(LO$+CHR$(0))+256*ASC(HI$+CHR$(0))" BLOCKS FREE
280 GOSUB730:GOSUB740:POKE140,0:RETURN
290 REM *** NOMBRE DE ENTRADA ***
300 PRINT!cr ab![F1] SALIDA!11 es![F7] DIRECTORIO"
310 PRINT!cr ab!NOMBRE DEL PROGRAMA? "CU$;:POKEB,0:F$=""
320 KQ=PEEK(KL):GETA$:IFA$=""ANDKQ=64THEN320
330 IFKQ=F1ORKQ=F7THENPRINTCHR$(20):RETURN
340 IFA$=CHR$(20)ANDF$=""THEN320
350 IFA$=CHR$(13)ANDF$<>""THENPRINTCHR$(20):RETURN
360 IFA$=CHR$(13)ANDF$=""THEN320
370 IFA$=CHR$(20)THENPRINTCHR$(20)A$CU$:
    F$=LEFT$(F$,LEN(F$)-1):GOTO320
380 PRINTCHR$(20)A$CU$;:F$=F$+A$:GOTO320
390 REM *** PROTEGE UN PROGRAMA ***
400 PRINT"!clr!";
405 PRINTSPC(NS-1)"PROTEGE UN PROGRAMA":PRINTU$
410 GOSUB300:IFKQ=F1THENRETURN
420 IFKQ=F7THENGOSUB210:GOTO400
430 POKE679,1:GOSUB780:GOSUB660:T=18:S=1
440 GOSUB60:SYS828:A=PEEK(252):IFATHEN470
450 GOSUB80:IFTTHEN440
460 PRINT!cr ab!!red!EL PROGRAMA NO SE ENCUENTRA!blk!!cr ab!":
    GOTO410
470 IF(AAND64)THENPRINTF$:PRINT"YA ESTABA PROTEGIDO":
    GOSUB730:GOSUB740:GOTO510
480 P=PEEK(255):GOSUB60:PRINT#15,"B-P";2;P:PRINT#2,CHR$(AOR64);
490 GOSUB70:IFETHEN400
500 GOSUB730:PRINTF$:PRINT"YA ESTA PROTEGIDO":GOSUB740
510 POKE679,0:GOTO400
520 REM *** DESPROTEGE ***
530 PRINT"!clr!";
535 PRINTSPC(NS-2)"DESPROTEGE UN PROGRAMA":PRINTU$
540 GOSUB300:IFKQ=F1THENRETURN
550 IFKQ=F7THENGOSUB210:GOTO530
560 POKE679,1:GOSUB780:GOSUB660:T=18:S=1
570 GOSUB60:SYS828:A=PEEK(252):IFATHEN600
580 GOSUB80:IFTTHEN570
590 PRINT!cr ab!!red!EL PROGRAMA NO SE ENCUENTRA!blk!!cr ab!":
    GOTO540
600 IF(AAND64)=.THENPRINTF$Q$"YA ESTABA DESPROTEGIDO":
    GOSUB730:GOSUB740:GOTO640
610 P=PEEK(255):GOSUB60:
    PRINT#15,"B-P";2;P:PRINT#2,CHR$(AAND135);
620 GOSUB70:IFETHEN530
630 GOSUB730:PRINTF$:PRINT"ESTA DESPROTEGIDO":GOSUB740
640 POKE679,0:GOTO530
650 REM *** APERTURA DE FILES ***
660 CLOSE2:CLOSE15:OPEN15,8,15,"I0":OPEN2,8,2,"#"
```

```
670 INPUT#15,E,EM$,ET,ES
680 IFE=73ORE=26THENGOSUB730:GOTO710
690 IFETHENPRINT"!red!ERROR:"
    PRINTE","EM$","ET","ES"!blk!":GOSUB730:END
700 RETURN
710 IFE=73THENPRINT"EL DISCO ESTA PROTEGIDO":GOSUB740:RETURN
720 PRINT"REMUEVA LA CINTA PROTECTORA":GOSUB740:RETURN
730 CLOSE2:CLOSE15:RETURN
740 PRINT"!rvs on!!cr ab!OPRIMA UNA TECLA!cr ab!":POKEKB,0
750 KQ=PEEK(KL):GETA$:IFA$=""ANDKQ=64THEN750
760 RETURN
770 REM *** ALMACENA NOMBRE ***
780 IFLEN(F$)<16THENF$=F$+CHR$(160):GOTO780
790 FORI=1TOLEN(F$):POKE683+I,ASC(MID$(F$,I,1)):
    NEXT:POKE700,0:RETURN
800 REM *** PROTEGE EL DISCO ENTERO ***
810 PRINT"!clr!"SPC(NS-2)"PROTEGE TODO EL DISCO":PRINTU$
820 PRINT!cr ab!INSERETE EL DISCO EN EL DRIVE"
822 PRINT"[F1]!8 es!ABORTA":FORD=1TO900:NEXT:GOSUB740
830 IFKQ=F1THENRETURN
840 GOSUB660:T=18:S=0:GOSUB60:PRINT#15,"B-P 2 2"
850 PRINT#2,CHR$(66);:PRINT#15,"B-P 2 166":GOSUB70:IFETHEN810
860 GOSUB730:GOSUB660:GOSUB730
870 PRINT"EL DISCO ESTA PROTEGIDO"
875 PRINT"PARA LA ESCRITURA":GOSUB740:RETURN
880 *** BORRA PROGRAMAS ***
890 PRINT"!clr!";
895 PRINTSPC(NS)"BORRA UN PROGRAMA":PRINTU$
900 GOSUB300:IFKQ=F1THENRETURN
910 IFKQ=F7THENGOSUB210:GOTO890
920 INPUT"ESTA UD SEGURO  (S-N)  N!3 cr iz!";A$:
    IFA$<>"S"THEN890
930 GOSUB660:PRINT#15,"S0:"+F$:INPUT#15,E,EM$,ET,ES:
    IFE>1THENGOSUB680:GOTO890
940 GOSUB730:IFET=.THENGOSUB975:GOSUB740:GOTO890
950 IFET>1THENPRINTET;EM$:GOSUB740:GOTO890
960 PRINTF$:PRINT"FUE BORRADO DEL DISCO":GOSUB740:GOTO890
970 REM *** DESPROTEGE EL DISCO ***
975 PRINT!cr ab!!red!EL PROGRAMA ESTA PROTEGIDO O NO SE"
976 PRINT"ENCUENTRA EN EL DISCO!blk!":RETURN
980 PRINT"!clr!"SPC(NS-3)" DESPROTEGE EL DISCO":PRINTU$
990 PRINT!cr ab!INSERTE EL DISCO EN EL DRIVE"
992 PRINT"[F1]!8 es!ABORTA":FORD=1TO900:NEXT:GOSUB740
1000 IFKQ=F1THENRETURN
1010 GOSUB660:T=18:S=0:GOSUB60
1020 PRINT#15,"M-W";CHR$(1);CHR$(1);CHR$(1);CHR$(65);
1030 PRINT#15,"B-P 2 2":PRINT#2,CHR$(65),:PRINT#15,"B-P 2 166"
1040 PRINT#2,CHR$(65),:GOSUB70:IFETHEN980
1050 GOSUB60:GOSUB730:PRINT"EL DISCO ESTA DESPROTEGIDO"
1060 GOSUB740:RETURN
1070 REM *** MENU PRINCIPAL ***
1080 PRINT"!clr!!blk!!cr ab!"SPC(NS)"<<<PROTECTOR>>>"
1085 PRINT!2cr ab!"
1090 PRINT!cr ab!!5 esp!1) DIRECTORIO
1100 PRINT"!5 esp!2) PROTEGE TODOS LOS PROGRAMAS
1110 PRINT"!5 esp!3) PROTEGE UN PROGRAMA
1120 PRINT"!5 esp!4) DESPROTEGE UN PROGRAMA
1130 PRINT"!5 esp!5) PROTEGE EL DISCO
1140 PRINT"!5 esp!6) DESPROTEGE EL DISCO
1150 PRINT"!5 esp!7) BORRA UN PROGRAMA
1160 PRINT"!5 esp!8) FIN
1165 PRINT!cr ab!"
1170 PRINT!2cr ab!!8 esp!SELECCIONE SU OPERACION"
1180 GETA$:IFA$=""THEN1180
1190 C=VAL(A$):IFC<1ORC>8THEN1180
1200 ONCGOSUB210,100,400,530,810,980,890,1220
1210 GOTO1080
1220 CLOSE2:CLOSE15:END
1230 REM *** VARIABLES Y LM ***
1240 POKE679,0:POKE140,0:CU$=CHR$(31)+CHR$(161)+CHR$(144)
1250 CP=0:FORQZ=828TO1006:READZQ:CP=CP+ZQ:POKEQZ,ZQ:NEXT
1255 IFCP<>26645THENPRINT"ERROR EN DATAS LINEAS 1260-1390.
     CHEQUEE VALORES":STOP
1256 CP=0:RETURN
1260 DATA 169,000,141,168,002,133,253,
          162,002,032,198,255,032
1270 DATA 228,255,141,169,002,032,228,
          255,141,170,002,230,253
1280 DATA 230,253,032,228,255,133,252,
          165,253,133,255,032,228
1290 DATA 255,032,228,255,230,253,230,
          253,160,000,032,228,255
1300 DATA 230,253,153,189,002,200,024,
          192,016,144,242,160,243
1310 DATA 032,228,255,230,253,165,144,
          240,003,141,168,002,200
1320 DATA 208,241,165,252,208,007,173,
          168,002,208,022,240,191
1330 DATA 165,140,240,003,032,199,003,
          173,167,002,240,003,032
1340 DATA 175,003,173,168,002,240,171,
          032,204,255,096,160,000
1350 DATA 185,172,002,240,008,217,189,
          002,208,007,200,208,243
1360 DATA 238,168,002,096,169,000,133,
          252
1370 DATA 096,160,000,185,189,002,240,
          006,032,210,255,200,208
1380 DATA 245,165,252,041,064,240,010,
          169,144,032,210,255,169,060,
          032,210,255,169
1390 DATA 013,032,210,255,169,144,032,
          210,255,96
1395 END
```

SET DEL MICRO-PROCESADOR 6510

Finalizamos la descripción del set de instrucciones del micro del Drean Commodore 64.



LSR Traslada el contenido del acumulador o memoria un bit hacia la derecha

Operación: 0→[7|6|5|4|3|2|1|0]→C

N	Z	C	I	D	V
0	✓	✓	/	/	/

modo de direccionamiento	mnemotécnico	CO	Nro. de bytes	Nro. de ciclos
Acumulador	LSR A	4A	1	2
Página Cero	LSR Oper	46	2	5
Página Cero, X	LSR Oper, X	56	2	6
Absoluta	LSR Oper	4E	3	6
Absoluta, X	LSR Oper, X	5E	3	7

NOP No opera durante 2 ciclos

Operación: no opera

N	Z	C	I	D	V
/	/	/	/	/	/

modo de direccionamiento	mnemotécnico	CO	Nro. de bytes	Nro. de ciclos
Implícito	NOP	EA	1	2

ORA Realiza OR lógica entre memoria y acumulador

Operación: (A) V (M)→A

N	Z	C	I	D	V
✓	✓	/	/	/	/

modo de direccionamiento	mnemotécnico	CO	Nro. de bytes	Nro. de ciclos
Inmediato	ORA # Oper	09	2	2
Página Cero	ORA Oper	05	2	3
Página Cero, X	ORA Oper, X	15	2	4
Absoluta	ORA Oper	ØD	3	4
Absoluta, X	ORA Oper, X	1D	3	4
Absoluta, Y	ORA Oper, Y	19	3	4
(Indirecta, X)	ORA (Oper, X)	01	2	6
(Indirecta), Y	ORA (Oper), Y	11	2	5

PHA Pone el contenido del acumulador en el stack

Operación: A↓

N	Z	C	I	D	V
/	/	/	/	/	/

modo de direccionamiento	mnemotécnico	CO	Nro. de bytes	Nro. de ciclos
Implícito	PHA	48	1	3

PHP Pone el contenido del registro de estado (SR) en el stack

Operación: P↓

N	Z	C	I	D	V
/	/	/	/	/	/

modo de direccionamiento	mnemotécnico	CO	Nro. de bytes	Nro. de ciclos
Implícito	PHP	08	1	3

PLA Toma desde el stack el valor del acumulador

Operación: A ↑ — N Z C I D V: √ √ / / / /

modo de direccionamiento	mnemotécnico	CO	Nro. de bytes	Nro. de ciclos
Implícito	PLA	68	1	4

PLP Toma desde el stack el estado del procesador.

Operación: P ↑ — N Z C I D V: desde el stack

modo de direccionamiento	mnemotécnico	CO	Nro. de bytes	Nro. de ciclos
Implícito	PLP	28	1	4

ROL Rota el contenido del acumulador o de memoria un bit hacia la izquierda

Operación: 7 6 5 4 3 2 1 0 ← C — N Z C I D V: √ √ / / /

modo de direccionamiento	mnemotécnico	CO	Nro. de bytes	Nro. de ciclos
Acumulador	ROL A	2A	1	2
Página Cero	ROL Oper	26	2	5
Página Cero, X	ROL Oper, X	36	2	6
Absoluta	ROL Oper	2E	3	6
Absoluta, X	ROL Oper, X	3E	3	7

ROR Rota el contenido del acumulador o memoria un bit hacia la derecha

Operación: C → 7 6 5 4 3 2 1 0 — N Z C I D V: √ √ √ / / /

modo de direccionamiento	mnemotécnico	CO	Nro. de bytes	Nro. de ciclos
Acumulador	ROR A	6A	1	2
Página Cero	ROR Oper	66	2	5
Página Cero, X	ROR Oper, X	76	2	6
Absoluta	ROR Oper	6E	3	6
Absoluta, X	ROR Oper, X	7E	3	7

RTI Retorna desde interrupción

Operación: P↑ PC↑ — N Z C I D V: desde el stack

modo de direccionamiento	mnemotécnico	CO	Nro. de bytes	Nro. de ciclos
Implícito	RTI	40	1	6

RTS Retorna desde subrutina

Operación: P↑, PC + 1→PC — N Z C I D V: / / / / / /

modo de direccionamiento	mnemotécnico	CO	Nro. de bytes	Nro. de ciclos
Implícito	RTS	60	1	6

SBC Resta los contenidos de memoria y acumulador con carry

Operación: (A) – (M)-C→A — N Z C I D V: √ √ √ / / √

modo de direccionamiento	mnemotécnico	CO	Nro. de bytes	Nro. de ciclos
Inmediato	SBC # Oper	F9	2	2
Página Cero	SBC Oper	E5	2	3
Página Cero, X	SBC Oper, X	F5	2	4
Absoluta	SBC Oper	ED	3	4
Absoluta, X	SBC Oper, X	FD	3	4
Absoluta, Y	SBC Oper, Y	F9	3	4
(Indirecta, X)	SBC (Oper, X)	E1	2	6
(Indirecta), Y	SBC (Oper), Y	F1	2	5

SEC Setea el flag de carry

Operación: 1→C

N	Z	C	I	D	V
/	/	1	/	/	/

modo de direccionamiento	mnemotécnico	CO	Nro. de bytes	Nro. de ciclos
Implícito	SEC	38	1	2

SED Setea modo decimal

Operación: 1→D

N	Z	C	I	D	V
/	/	/	/	1	/

modo de direccionamiento	mnemotécnico	CO	Nro. de bytes	Nro. de ciclos
Implícito	SED	F8	1	2

SEI Setea flag de interrupción

Operación: 1→I

N	Z	C	I	D	V
/	/	/	1	/	/

modo de direccionamiento	mnemotécnico	CO	Nro. de bytes	Nro. de ciclos
Implícito	SEI	78	1	2

STA Almacena el contenido del acumulador en memoria

Operación: (A)→M

N	Z	C	I	D	V
/	/	/	/	/	/

modo de direccionamiento	mnemotécnico	CO	Nro. de bytes	Nro. de ciclos
Página Cero	STA Oper	85	2	3
Página Cero, X	STA Oper, X	95	2	4
Absoluta	STA Oper	8D	3	4
Absoluta, X	STA Oper, X	9D	3	5
Absoluta, Y	STA Oper, Y	99	3	5
(Indirecta, X)	STA (Oper, X)	81	2	6
(Indirecta), Y	STA (Oper), Y	91	2	6

STX Almacena el contenido del registro X en memoria

Operación: (X)→M

N	Z	C	I	D	V
/	/	/	/	/	/

modo de direccionamiento	mnemotécnico	CO	Nro. de bytes	Nro. de ciclos
Página Cero	STX Oper	86	2	3
Página Cero, Y	STX Oper, X	96	2	4
Absoluta	STX Oper	8E	3	4

STY Almacena el contenido del registro Y en memoria

Operación: (Y)→M

N	Z	C	I	D	V
/	/	/	/	/	/

modo de direccionamiento	mnemotécnico	CO	Nro. de bytes	Nro. de ciclos
Página Cero	STY Oper	84	2	3
Página Cero, X	STY Oper, X	94	2	4
Absoluta	STY Oper	8C	3	4

TAX Transfiere el contenido del acumulador al registro X

Operación: (A)→X

N	Z	C	I	D	V
√	√	/	/	/	/

modo de direccionamiento	mnemotécnico	CO	Nro. de bytes	Nro. de ciclos
Implícito	TAX	AA	1	2

TAY Transfiere el contenido del acumulador al registro Y

Operación: (A)→X

N	Z	C	I	D	V
√	√	/	/	/	/

modo de direccionamiento	mnemotécnico	CO	Nro. de bytes	Nro. de ciclos
Implícito	TAY	AB	1	2

TSX Transfiere el stack pointer al registro X

Operación: (S)→X

N	Z	C	I	D	V
√	√	/	/	/	/

modo de direccionamiento	mnemotécnico	CO	Nro. de bytes	Nro. de ciclos
Implícito	TSX	BA	1	2

TXA Transfiere el contenido del registro X al acumulador

Operación: (X)→A

N	Z	C	I	D	V
√	√	/	/	/	/

modo de direccionamiento	mnemotécnico	CO	Nro. de bytes	Nro. de ciclos
Implícito	TXA	8A	1	2

TXS Transfiere el contenido del registro X al stack pointer

Operación: (X)→S

N	Z	C	I	D	V
√	√	/	/	/	/

modo de direccionamiento	mnemotécnico	CO	Nro. de bytes	Nro. de ciclos
Implícito	TXS	9A	1	2

TYA Transfiere el contenido del registro Y al acumulador

Operación: (Y)→A

N	Z	C	I	D	V
√	√	/	/	/	/

modo de direccionamiento	mnemotécnico	CO	Nro. de bytes	Nro. de ciclos
Implícito	TYA	98	1	2

CLASIFICANDO DATOS

Si necesitamos clasificar datos, debemos realizar nuestros propios algoritmos. Existen en la actualidad una gran cantidad de ellos, desde el más lento hasta el super veloz. Nosotros explicaremos algunos de ellos para que puedan utilizarlos a la hora de ordenar información.

El proceso de clasificar (sort) objetos es, sin duda alguna, la tarea más frecuente que realizamos. Ordenar el dinero de mayor a menor valor, insertar en el lugar correcto un nuevo apellido a nuestra agenda telefónica, son algunos de los ejemplos.

En computación este procedimiento es aún más frecuente. Todos los medianos y grandes equipos disponen de rutinas o programas desarrollados exclusivamente para clasificar. Ellos utilizan algoritmos que garantizan un rápido y seguro "sort" de la información.

Por supuesto la elección del algoritmo, es decir el método elegido para resolver nuestro problema, dependerá del tiempo que éste tarde para ordenar de mayor a menor (o viceversa) una determinada cantidad de datos.

Lamentablemente esta tarea no viene incluida dentro de las computadoras hogareñas, como la Drean Commodore. Por ello debemos implementar nuestras propias rutinas de ordenamiento.

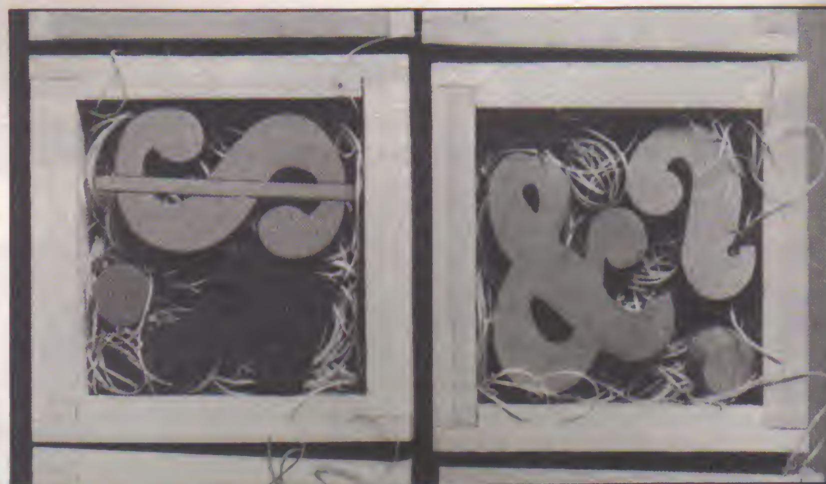
Para explicar dos de los tantos métodos de clasificación utilizaremos los vectores como estructura de datos. En el número anterior ya hicimos referencia a ellos. Les pedimos, a los que deben refrescar los conocimientos, que lean aquélla antes de iniciar la lectura de ésta.

Método 1:

Supongamos que los elementos del vector V son 3,5,7,1,9 y 2. Nuestro objetivo es ordenar sus elementos de menor a mayor.
Este método consiste en tomar un elemento del vector (al cual se lo llama base) y compararlo con los restantes. En caso de que encontremos un elemento que sea menor a la base, realizamos el intercambio de esos dos elementos; el base pasa al lugar que ocupa el menor y el menor al lugar que ocupa el base.
Para nuestro ejemplo realicemos el siguiente análisis:

Tomamos el primer elemento (3) y lo comparamos con el segundo (5). Si el primer elemento es menor o igual que el segundo no realizamos ninguna operación. Si, en cambio, el primer elemento fuese mayor que el segundo, debemos intercambiarlos de lugar.
En nuestro caso 3 es menor que 5, por lo que no realizamos la operación.
A continuación seguimos comparando el primer elemento con, ahora, el tercero (7). Como sucede lo mismo que en el caso anterior, no realizamos ninguna operación.

Luego comparamos el primero con el cuarto (1). Como aquí sí 3 es mayor que 1, debemos intercambiarlos, por lo que el orden parcial de nuestro vector es 1,5,7,3,9 y 2. El elemento base es, ahora, el 1.
Seguimos comparando el primer elemento (que ahora es 1) con el quinto elemento. Como 1 es menor que 9 no operamos.
Finalmente comparamos el base con el último (2). Mismo caso que el anterior. Es decir que luego de la primera pasada sobre nuestro vector los elementos son: 1,5,7,3,9 y 2.
Aquí ya se puede distinguir una de las características de este método. En la primera pasada se determina el menor elemento.
Seguimos la clasificación tomando como base el segundo elemento (5) ya que el primero es el menor. Por ello lo comparamos con el tercero. Como 5 es menor que 7 no operamos. Luego



comparamos el base con el cuarto elemento (3). Como 5 es mayor que 3 realizamos el intercambio, quedando el vector 1,3,7,5,9 y 2. El base es ahora 3 y la siguiente comparación debemos hacerla con el quinto elemento (9), en donde no operamos por ser 3 menor que 9.
Finalmente comparamos 3 con el último (2) en donde debemos intercambiarlos por ser 3 mayor que 2.
En la segunda pasada el vector queda en 1,2,7,5,9 y 3.
Continuamos tomando como base al

tercer elemento (7) y lo comparamos con el cuarto (5). Como es mayor los intercambiamos quedando el vector: 1,2,5,7,9 y 3. El elemento base pasó a ser el 5. Seguimos comparando el base con el quinto elemento (9). Es menor, no operamos.
Terminamos comparando el base con el último (2) realizando, por ser mayor el base, el intercambio.
El vector queda luego de la tercer pasada: 1,2,3,7,9 y 5.
Ahora tomamos como base el cuarto elemento (7). Lo comparamos con el quinto. Como es menor no operamos.
Finalmente lo comparamos con el último, en donde debemos realizar el intercambio.
En la cuarta pasada el vector queda: 1,2,3,5,7 y 9.
Ustedes pensarán que terminó el proceso pero, lamentablemente, no es así. Como esto es un algoritmo que será ejecutado por una computadora, ésta lo respeta hasta el final. Y este algoritmo sigue comparando los elementos a pesar de que están ordenados.
Por ello se toma como base el quinto elemento (7) y se lo compara con el último (9). Como es menor se deja todo igual finalizando el procedimiento.
Como ya habrán notado, la "contra" que tiene este proceso es que se sigue comparando con elementos que ya están ordenados.
Otro de los "peros" de este método es el tiempo que tarda. Para este caso se debieron hacer 5 pasadas para ordenar el vector. En general si un vector tiene n elementos, éste estará ordenado luego de la n-1 pasadas (imaginen vectores de 100 ó más elementos).
Como ejemplo final de este método, les dejamos el listado 1, el cual ordena un vector cuyos elementos son ingresados por teclado.

Listado 1

```
1 REM METODO NRO1
5 DIM V(100)
10 INPUT"CANTIDAD DE ELEMENTOS";N%
20 IFN%>100THEN10
30 FORI=1TON%:PRINT"INGRESE ELEMENTO
   NRO";I
40 INPUTV(I):NEXT
50 REM COMIENZA CLASIFICACION
60 FORI=1TON%-1
70 FORJ=I+1TON%
80 IFV(I)<=V(J)THEN100
90 AUX=V(I):V(I)=V(J):V(J)=AUX:REM
   INTERCAMBIO DE ELEMENTOS
100 NEXTJ
110 NEXTI
120 FORI=1TON%:PRINTV(I);:NEXT
130 END
```

Listado 2

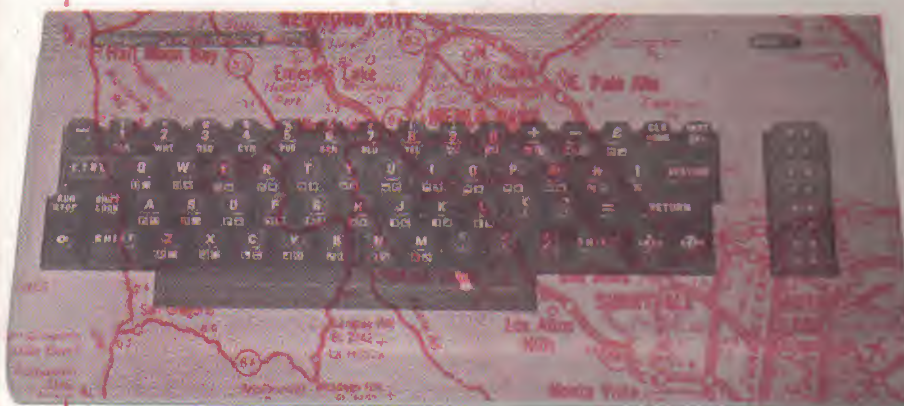
```
1 REM METODO NRO2
5 DIM V(100):F=0
10 INPUT"CANTIDAD DE ELEMENTOS";N%
20 IFN%>100THEN10
22 FORI=1TON%:PRINT"INGRESE ELEMENTO";I
25 INPUTV(I):NEXT
30 FORJ=1TON%
40 FORI=1TON%-1
50 IFV(I)<=V(I+1)THEN100
60 AUX=V(I)
70 V(I)=V(I+1)
80 V(I+1)=AUX
90 F=1
100 NEXTI
110 IFF=0THEN130
120 F=0:NEXTJ
130 FORI=1TON%
140 PRINTV(I);:NEXT
150 END
```

Método 2:

Este método, a diferencia del anterior, no realiza comparaciones innecesarias.
Supongamos que los elementos del vector son los mismos que antes, es decir 3,5,7,1,9 y 2.
La filosofía de este método es la siguiente: se compara el primero con el segundo. Si es mayor se intercambia. Si es menor no se opera y se compara el segundo con el tercero. Si es menor no se opera. Si es mayor se intercambian y se compara con el que sigue y así sucesivamente.
Este método también se lo conoce con el nombre de "burbuja".
Los elementos se van comparando de a pares sucesivos.
En nuestro ejemplo sería, con el vector inicial en 3,5,7,1,9 y 2, como:
Comparamos el primero con el segundo. Como 3 es menor que 5 no operamos y comparamos el segundo con el tercero (7). Como 5 es menor que 7 no operamos y pasamos a comparar el tercero con el cuarto (1). Como 7 es mayor que 1 los intercambiamos, quedando nuestro vector 3,5,1,7,9,2.
Seguimos comparando el cuarto (ahora es 7) con el quinto (9). Como es menor no operamos y pasamos a comparar el quinto con el último (2). Como 9 es mayor que 2 los intercambiamos quedando el vector: 3,5,1,7,2,9. Noten que el elemento más grande quedó, en la primera pasada, al final de todo.
Y ahora viene la diferencia. Como el elemento más grande quedó en la última posición, debemos repetir el proceso sin incluirlo. Es como repetir el caso para un vector cuyos elementos son el 3,5,1,7,2.
Al igual que antes determinaremos el mayor (en este caso 7) quedándonos el vector parcial 3,5,1,2. Repetimos el próceso, en donde sacaremos el 5 como el más grande, quedando 3,1,2.
Continuamos así hasta llegar a un vector con un elemento. Finalmente el vector quedará ordenado.
El programa que ejemplifica este algoritmo corresponde al listado 2.

MAPA DE MEMORIA

Continuamos describiendo la subrutina que se encarga de tomar un caracter desde el Basic y colocarlo en el acumulador.



En el número anterior comenzamos a describir una de las subrutinas más importantes que utiliza el sistema operativo de la C-64: CHRGET. Además, imprimimos su listado en Assembler.

Esta rutina se puso en la página cero debido a que su ejecución es mucho más rápida. Básicamente utiliza un puntero que ella va incrementando cada vez que se ejecuta.

Este puntero forma parte de la rutina por lo que se dice que se automodifica.

Podemos dividirla en dos partes: Una rutina, denominada CHRGET (character get, toma de caracter) la cual se llama con JSR $0073 y cuya función es tomar un byte del texto basic (o del buffer de entrada) y depositarlo en el acumulador.

La segunda, denominada CHRGOT, se llama a traves de la instrucción JSR $0079. Esta ultima permite que el último byte tomado permanezca en el acumulador. De esta manera se puede volver a leer.

La rutina CHRGET indica, seteando determinados flags del registro de estado del microprocesador, si el byte tomado fue un digito, si fue fin de sentencia u otro tipo de caracter.

Por ejemplo si el byte leido corresponde al codigo ASCII de los digitos comprendidos entre 0 y 9, entonces el flag de carry sera puesto a cero. Por el contrario sera seteado (puesto a uno). Si el byte tomado es un fin de sentencia (cero) o dos puntos (:) entonces el flag zero sera seteado. De lo contrario sera puesto a cero.

Otro punto importante de resaltar es que esta rutina ignora los espacios en blanco. De encontrarse con uno de ellos vuelve a tomar el siguiente caracter.

Hemos desensamblado esta rutina para que puedan analizarla en la tabla 1.

Dirección $008B-$008F (139-143):

Estas direcciones tienen el valor de un número en punto flotante representado por cinco bytes el cual es determinado por la función RND.

Cuando se llama a una función RND(X), el valor numérico de X no afecta al número retornado por ella (si interviene en el signo). Si X=0, la función RND genera un número aleatorio desde el chip del timer. Si X es un número positivo, el número se genera desde una serie aritmética. Si el número es negativo, el número que se genera es una mezcla de la representación en punto flotante del argumento.

La variable X se conoce con el nombre de "semilla". Dando un valor particular de ésta se generan una serie de números pseudo aleatorios.

Esto significa que tarde o temprano la serie se repetirá nuevamente. Una de las maneras de evitar ésto es usar como semilla la variable TI de la siguiente manera: RND(-TI). Otra de las expresiones que genera más números aleatorios es RND (RND (0)).

Dirección $0090-$00FF (144-255):

Esta es el área de almacenamiento del Kernal (sistema operativo de la C-64). Cuando se prende la coimputadora esta area es, primeramente, llenada con ceros para luego inicializarse con valores almacenados en ROM.

Pasamos pues a describir cada una de las direcciones que las constituyen:

$0090 (144): La rutina que se encarga de abrir canales de entrada-salida o ejecuta funciones de entrada/salida chequea y escribe en esta dirección. Ella representa la variable de estado ST. Por ello el basic no permite asignaciones de esta variable por parte del programador.

La siguiente tabla representa los valores de ST acorde a si el periférico externo es cassette o un dispositivo serie, como ser drive:

Cassette

bit2 (valor 4)	= Short Block
bit3 (valor 8)	= Long Block
bit4 (valor 16)	= Unrecoverable error (Read), mismatch
bit5 (valor 32)	= Checksum error
bit6 (valor 64)	= Enf Of File
bit7 (valor 128)	= Enf Of Tape

Dispositivo serie:

bit0 (valor 1)	= Time Out (write)
bit1 (valor 2)	= Time Out (read)
bit6 (valor 64)	= EOI (End Or Identify)
bit7 (valor 128)	= Device not Present

Probablemente el más utilizado es el bit

6 (End of File) cuando se utiliza la sentencia GET para leer byte por byte desde un archivo. Si la sentencia IF ST AND 64 es verdadera, entonces se llegó al EOF.

$0091 (145):
Esta dirección es escrita cada 1/60 segundos durante la ejecución de la rutina IRQ la cual se encarga de leer el teclado. Aquí se pone el código de la tecla correspondiente a la última fila. Esta fila contiene la tecla de STOP. Principalmente esta dirección es utilizada para detectar cuando esa tecla fue presionada y, también, para ver qué tecla perteneciente a esa fila fue presionada. De acuerdo a la tecla que se oprima, esta dirección contendrá los siguientes posibles valores:

$FF (255): no se oprimió ninguna tecla
$FE (254): se oprimió la tecla del 1
$FD (253): se oprimió la tecla
$FB: (251): se oprimió la tecla CTRL
$F7 (247): se oprimió la tecla del 2
$EF (239): se oprimió la barra espaciador
$DF (223): se oprimió la tecla de commodore
$BF (191): se oprimió la tecla Q
$7F (127): se oprimió la tecla de STOP

$0092 (146): El contenido de esta dirección se utiliza para realizar ajustes en la lectura de cinta.

$0093 (147): El contenido de esta dirección se utiliza como flag para indicarle al sistema operativo si debe ejecutar un comando LOAD o VERIFY.

$0094 (148): El contenido de esta dirección se utiliza para indicar que un caracter ha sido puesto en el buffer de salida y está esperando ser enviado hacia el bus serie.

$0095 (149): Aquí está el caracter esperando para ser enviado. Un valor de $FF en esta dirección, indica que no hay caracteres esperando.

$0096 (150): Se utiliza para almacenar el número de bloques de sincronización del cassette.

$0097 (151): Esta dirección se utiliza para almacenar temporalmente el contenido del registro X. Ello lo hace la rutina que toma y pone un caracter.

$0098 (152): Aquí se almacena el número de archivos abiertos actualmente. El máximo número de archivo que se puede abrir, al mismo tiempo es de diez. El contenido de esta dirección es utilizado como un índice dentro de una tabla que contiene el número de archivo lógico, número de dispositivo y el número de dirección secundaria.
La sentencia CLOSE disminuye el contenido de esta dirección quitando, además, la información de la tabla antes mencionada.
Por el contrario, la sentencia OPEN aumenta el contenido de esta dirección y pone la información necesaria en la tabla anterior. La rutina CLALL (ver en este mismo número la descripción de ésta) pone 0 en esta dirección.

$0099 (153):
El contenido de esta dirección se utiliza para indicar que dispositivo se está utilizando como entrada.
Generalmente el valor aquí almacenado es cero, ya que el ingreso de datos se hace a través del teclado.
La rutina CHKIN (descripta en la nota "Las subrutinas del Drean Commodore 64" de este número) la utiliza para almacenar el número de dispositivo cuyo archivo asociado fue definido como entrada.

$009A (154):
En esta dirección se almacena el número de dispositivo que se está utilizando como salida.
Generalmente ese valor es 3, el cual significa que se utiliza la pantalla como periférico de salida.
Este valor puede ser modificado por la rutina CHKOUT, la cual almacena aquí el número de dispositivo cuyo archivo asociado fue definido como salida.

$009B (155):
Esta dirección se utiliza durante la transmisión de información desde el cassette (paridad).

$009C (156):
Esta dirección se utiliza como flag para indicar si se ha recibido en forma completa un byte (dato) desde el cassette o cuándo se recibió en forma parcial.

$009D (157):
Esta dirección es utilizada como flag por la rutina SETMSG.
De acuerdo al contenido de ésta, el sistema operativo imprimirá los mensajes de error o los de control.
Un valor de 192 ($C0) indicará que se deben imprimir ambos tipos de mensajes.
Un valor de 128 ($80) indicará que sólo se deben imprimir los mensajes de control.
Un valor de 64 indicará que se deben imprimir, sólo, los mensajes de error.
Un valor de 0 suprime todos los mensajes.

$009E (158):
Esta dirección se utiliza durante la transmisión de bloques de datos.

$009F (159):
Se utiliza en conjunto con la anterior dirección.

$00A0-$00A2 (160-162):
Los contenidos de estas tres direcciones son modificadas 60 veces por segundo. Se utilizan como un reloj de software, contando, en unidades de 1/60, el tiempo que transcurrió desde que se activó la computadora.
El contenido de la dirección $A2 (162) es incrementado cada .0166 segundos (16.6 mseg). La dirección $A1 (161) es incrementada (su contenido) cada 4.22 segundos. Finalmente, la dirección $A0 (160) es incrementada cada, aproximadamente, 183 minutos.
Luego de completar las 24 horas, estas direcciones son puestas a cero.
Estas direcciones son utilizadas por la variable reservada TI y TI$.

$00A3-$00A4 (163-164):
Estas direcciones se utilizan como área de almacenamiento temporal por las rutinas de cinta y entradas/salidas.

$00A5 (165):
Esta dirección se utiliza para contar el número de caracteres de sincronización enviados antes del dato actual dentro del bloque de información.

$00A6 (166):
Esta dirección es utilizada para contar el número de caracteres que deben ser leídos o escritos desde el buffer de cassette.
Si se está realizando un proceso de escritura en la cinta (un SAVE), no se envían datos hasta que no se "vacíe" ese buffer, que tiene la capacidad de almacenar 192 bytes.

$00A7 (167):
Esta dirección se utiliza para almacenar temporalmente cada bit que se recibe desde el bus serie.

$00A8 (168):
Esta dirección se utiliza para contar el número de bits que se están recibiendo desde el bus serie.

$00A9 (169):
Esta dirección es utilizada como flag para indicar si hubo bit de inicio o de fin, durante el proceso de transferencia de información desde el dispositivo serie hacia la computadora.

LAS VENTAJAS DEL DRIVE 1541

Este periférico es ideal para trabajar con archivos relativos. Dispone de mayor densidad para el almacenamiento de datos que el Datassette y su velocidad de operación es muy superior.

A diferencia del Datassette (unidad de cinta para la C-64), el Disk Drive 1541 dispone de mayor densidad para el almacenamiento de datos, ya que trabaja con diskette en lugar de cinta magnética. Además, su velocidad de operación es muy superior con respecto al anterior. Pero, si debemos comparar la 1541 frente a otras disketeras (como ser la nueva 1571), la velocidad de operación de ésta es muy inferior a las demás.
Otra de las ventajas que se presentan es la de poder trabajar con archivos relativos. Con este tipo de organización podemos acceder a un determinado registro sin tener que pasar por los anteriores (como sucede en el caso de los secuenciales).
Otras de las características de este equipo, y creemos que es la más importante, es su gran fiabilidad.
Por otro lado, y como describiremos en futuras notas, es notable el "cuidado" que debe tenerse cuando se trabaja con archivos de acceso directo.
Antes mencionamos la relativa lentitud en la operación del Drive. Con esto queremos decir que la transferencia de información (ya sea desde o hacia el drive) es demasiado lenta.
Sin embargo podemos solucionar este problema si disponemos de un turbo disk o algún otro utilitario que aumente esa velocidad de transferencia.
También se le puede insertar a la computadora un cartucho denominado Fast Load. Este, al igual que los anteriores, aumenta la velocidad de lectura y/o escritura de datos, haciéndola operar aun más rápido.

DOS 5.1

D.O.S. son las siglas de Disk Operating System (sistema operativo de disco). Se encarga de administrar los recursos y la operación del drive.
Dispone de una serie de comandos para que el programador pueda operar cómodamente el drive. Por ejemplo, en modo normal, cargar el directorio del disco actual, se realiza a través del comando LOAD"$". Luego debemos tipear LIST para visualizarlo.
Además, todo programa en memoria será destruido.
Con el DOS, esta tarea se lleva a cabo con sólo oprimir la tecla "@$". De esta manera se imprimirá en pantalla el directorio del diskette sin destruir datos en memoria.
Otro ejemplo es cuando se debe cargar un programa desde el disco. Comúnmente, esto se realiza a través de LOAD"nombre del programa",8. Con el DOS esto se realiza como /nombre del programa.
Si se hubiese necesitado la opción ",1" (con lo que tuvimos que efectuar LOAD"nombre del programa",8,1), en DOS esto se realiza a través de %nombre del programa.
Si deseamos grabar programas en disco, el DOS dispone del comando "<-". Si debemos grabar sobre un programa ya existente, se utiliza la opción de reemplazo como "@ :nombre del programa".
Para inicializar la disketera, se dispone del comando "I". Este se envía a ella a través de "@I". Validar un disco (es decir recuperar espacio libre) se hace a través del comando "@V". A través de "@Q" se desactiva el DOS. Este se vuelve a activar a través de "@U1".
El sistema operativo de disco generalmente viene dentro del disco que acompaña la disketera. Además dentro de él se encuentran otros programas utilitarios que permiten realizar un chequeo general al drive, comprobar si escribe y lee información correctamente, etc.

DIRECCION		DESCRIPCION
DEC	HEX	
000	$000	Código CMD para buffer 0
001	$001	Código CMD para buffer 1
002	$002	Código CMD para buffer 2
003	$003	Código CMD para buffer 3
004	$004	Código CMD para buffer 4
006	$06-$07	Track y sector del buffer 0
008	$08-$09	Track y sector del buffer 1
010	$0A-$0B	Track y sector del buffer 2
012	$0C-$0D	Track y sector del buffer 3
014	$0E-$0F	Track y sector del buffer 4
030	$01E	Protección contra escritura
061	$03D	Número actual de drive
063	$03F	Número actual de buffer
586	$24A	Tipo de archivo
768	$300-$3FF	Buffer 0
1024	$400-$4FF	Buffer 1
1280	$500-$5FF	Buffer 2
1536	$600-$6FF	Buffer 3
1792	$700-$7FF	Buffer 4

UNIDAD DE DISCO PARA 16 Y 64

Características

La principal característica es que el drive es un dispositivo "inteligente". Este término se le atribuye a todo periférico que contiene en su interior un microprocesador.
En el caso del drive 1541, éste dispone de un micro 6502 (similar al que tiene la C-64) el cual se encarga de hacer operar correctamente a la unidad.
También dispone de un buffer de 2Kb, utilizado para el almacenamiento temporal de la información.

Su capacidad de almacenamiento es de 170Kb por disco, disminuyendo ésta un poco en caso de trabajar con archivos relativos o secuenciales.

En este caso, es decir si se trabaja con archivos de organización secuencial, se dispone de la opción "A", la cual permite insertar datos al final de un archivo secuencial ya creado. Por ejemplo OPEN2,8,3,"TEST,S,A" abrirá el archivo TEST y agregará datos al final de éste.
Otra de las posibilidades que se nos permite es la de poder programar el controlador del disco. Para ello disponemos de ciertos comandos, los cuales posibilitan grabar instrucciones en código máquina en determinadas direcciones de memoria.
Lo último descripto es muy útil cuando deseamos modificar el funcionamiento del drive logrando, entre otras cosas, mayor velocidad de operación.
Volviendo a los archivos, la Drean Commodore 64 dispone de una variable de estado (comunmente llamada ST) la cual nos permite saber cuándo se llegó al final de un archivo (EOF). En este caso el contenido de dicha variable será 64.

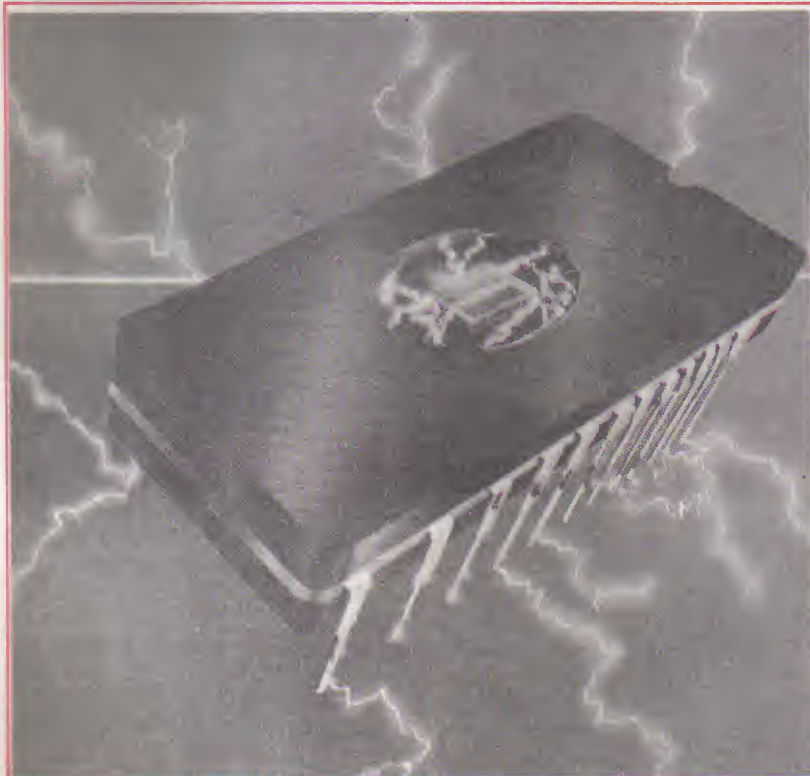
Configuración interna

Como mencionamos en otras notas, conocer las direcciones de memoria de un determinado equipo posibilitan el hecho de, primero, investigarlo para luego mejorar o cambiar algunos aspectos de su funcionamiento.
Por ejemplo, para proteger un disco contra la escritura podemos poner la correspondiente etiqueta de protección o bien escribir un determinado valor en un determinado lugar del disco que simula el primer hecho.
Como ustedes saben se nos permite cambiar el número de dispositivo de 8 a 9. Para ésto también existen dos formas. Una es por hardware, la cual no recomendamos ya que se debe abrir la unidad con todos los riesgos que ello implica.
La segunda es a través del software. Aquí debemos enviar a una determinada dirección de memoria (dentro de la 1541) el valor que representa el nuevo número de dispositivo.
Esta dirección es justamente la que le indica al sistema operativo del disco cuál es el número de drive actual.
La tabla 1 representa las direcciones de memoria del drive 1541 con su correspondiente significado.
Como verán hay determinadas direcciones que indican, por ejemplo, si un disco está protegido contra escritura, el tipo de archivo actual, número actual de drive y otras funciones.



LAS SUBRUTINAS DEL DREAN COMMODORE 64

Continuamos describiendo las subrutinas en Rom utilizadas por el sistema operativo de la C-64.



Hemos visto, en el número anterior, cómo acceder a las rutinas que utiliza el sistema operativo de la C-64. Habíamos dado ejemplos, usando como lenguaje el Assembler.
Antes de seguir describiendo las subrutinas restantes, relataremos cómo acceder a ellas pero utilizando el Basic en lugar del Assembler.
Para ello debemos describir algunas de las herramientas que dispone este lenguaje para trabajar con rutinas o programas en código de máquina.
La primera, y tal vez la más importante, es el comando SYS que transfiere el control a un programa en código máquina ubicado a partir de la dirección especificada. Por ejemplo, SYS49152 ejecutará un programa en código máquina almacenado a partir de la dirección decimal 49152 ($C000 hex).
La segunda son las direcciones en Ram utilizadas por el acumulador, registro X, registro Y y el registro de estado del microprocesador, necesarias para el intercambio de parámetros entre la rutina y el programa.
Estas son las direcciones 780, 781, 782 y 783, respectivamente.
Si, en Assembler, cargar el registro X con FF (hexadec.) se realizaba como LDX #$FF, aquí se hará como POKE781,255.
Habíamos dicho, además, que en caso de ocurrir un error el flag de carry es seteado (puesto a "1"). En Basic esto lo podemos detectar haciendo:

```
IFPEEK(783)AND1=1THEN
PRINT"ERROR":STOP
```

Con todas ellas podrán utilizar desde el Basic las rutinas, las cuales continuamos describiendo:

Nombre de la función: CHRIN

Función: Toma un caracter desde el canal de entrada.
Dirección de llamada: $FFCF (hex) 65487 (decimal).
Registros de comunicación: A
Rutina preliminar: OPEN, CHKIN
Error: 0 (ver también la rutina READST)
Requerimientos de stack: 7
Registros afectados: A,X
Descripción: Esta rutina toma un caracter desde el canal abierto como entrada por las rutinas OPEN y CHKIN. En caso de que no haya sido abierto otro canal como entrada (no se usa la rutina CHKIN), los datos se tomarán desde el teclado.
El byte leído desde el canal es colocado en el acumulador. El canal permanece abierto luego del acceso a la rutina.
El ingreso desde el teclado se realiza de la siguiente manera:
Primero se activa el cursor y se lo hace parpadear hasta que se oprima la tecla return. Todos los caracteres ingresados son almacenados en el buffer de entrada. Cada uno de ellos puede ser leído usando esta rutina. Por cada acceso a ella se lee un caracter hasta que se oprima return.
Pasos a seguir:
a) Desde el teclado:
1) Tomar un caracter a través de la rutina
2) Almacenarlo
3) Comprobar si es return
4) Si no es, ir al punto 1. Si es finalizar.
Ejemplo:

```
LEER:
LDY #$00; preparamos registro Y
JSR $FFCF; leemos un caracter
STA BYTE,Y; y lo almacenamos
INY; incrementamos índice
CMP #$0D; si no es return
BNE LEER; leemos el que sigue.
```

Desde el Basic podemos hacerlo como:

```
5 A$=""
10 SYS 65487
20 A$=A$+CHR$(PEEK(780))
30 IFPEEK(780)=13THENPRINT:
PRINTA$:STOP
40 GOTO 10
```

Observen que es posible utilizar esta rutina para analizar ingresos de datos sin

que aparezca el signo de pregunta de la sentencia INPUT.
b) Desde otro dispositivo:
1) Utilizar las rutinas OPEN y CHKIN para abrir el canal y ponerlo como entrada.
2) Acceder a la rutina.
3) Almacenar el dato leído.
El código de error 0, devuelto en el acumulador si el carry está en "1", significa que la rutina fue suspendida por la tecla STOP (se oprimió).

Nombre de la función: CHROUT
Función: Saca un caracter hacia el canal de salida.
Dirección de llamada: $FFD2 (hex) 65490 (decimal).
Registros de comunicación: A
Rutina preliminar: CHKOUT, OPEN
Error: 0 (ver también la rutina READST)
Requerimientos de stack: 8
Registros afectados: A
Descripción: Esta rutina envía un caracter hacia el canal actualmente abierto. Antes se debe acceder a las rutinas OPEN y CHKOUT para determinar el canal de salida.
Si éstas se omiten, los datos son enviados hacia la pantalla (dispositivo número 3).
En ambos casos el acumulador debe contener el código ASCII del caracter a enviar, antes de acceder a esta rutina.
Pasos a seguir:
1) Utilizar, si se requieren, las rutinas preliminares.
2) Cargar el dado dentro del acumulador.
3) Acceder a la rutina.
Ejemplo:
LDA #$41; ASCII de 'A'
JSR $FFD2; lo imprimimos en pantalla
Desde el Basic:
10 POKE780,65
20 SYS65490
30 STOP

Nombre de la función: CIOUT
Función: Transmite un byte sobre el bus serie.
Dirección de llamada: $FFA8 (hex) 65448 (decimal).
Registros de comunicación: A
Rutina preliminar: LISTEN, [SECOND]
Error: ver la rutina READST
Requerimientos de stack: 5
Registros afectados: Ninguno.
Descripción: Esta rutina se utiliza para enviar información (más precisamente un byte a la vez) hacia el dispositivo conectado sobre el bus serie.
Antes de llamar a esta rutina, se debe acceder la rutina LISTEN para determinar qué dispositivo recibirá la información. (Si el dispositivo necesita una dirección secundaria o comando, entonces se debe acceder, también, a la rutina SECOND).
Al igual que la rutina anterior (CHROUT), el acumulador contendrá el byte a transmitir. En caso de que el dispositivo no haya sido preparado para recibir los datos, se producirá timeout (fuera de tiempo).
Pasos a seguir:
1) Llamar a la rutina LISTEN (y si se necesita, a la SECOND).
2) Cargar el acumulador con el byte a transmitir.
3) Llamar a la rutina.
Ejemplo:
LDA #$41; cargamos acumulador con ASCII de "A".
JSR $FFAB; lo enviamos hacia el dispositivo sobre el bus serie (primero debemos llamar a las rutinas LISTEN y SECOND).

Nombre de la función: CINT
Función: Inicializa el editor de pantalla y el chip de video 6567.
Dirección de llamada: $FF81 (hex) 65409 (decimal).
Registros de comunicación: ninguno.
Rutina preliminar: ninguna.
Error: ninguno.
Requerimientos de stack: Y
Registros afectados: A,X,Y
Descripción: Esta rutina inicializa el chip de video 6567 y el editor de pantalla.
Pasos a seguir:
1) Acceder a esta rutina.
Ejemplo:
JSR $FF81
Desde el Basic:
10 SYS65409

Nombre de la función: CLALL
Función: Cierra todos los archivos.
Dirección de llamada: $FFE7 (hex) 65511 (decimal).
Registros de comunicación: ninguno.
Rutina preliminar: ninguna
Error: ninguno.
Requerimientos de stack: 11
Registros afectados: A,X
Descripción: Esta rutina cierra todos los archivos abiertos.
Pasos a seguir:
1) Acceder a esta rutina.
Ejemplo:
JSR $FFE7
Desde el Basic
10 SYS65511

Nombre de la función: CLOSE
Función: Cierra un archivo.
Dirección de llamada: $FFC3 (hex) 65475 (decimal)
Registros de comunicación: A
Rutina preliminar: ninguna.
Error: 0,240 (ver también la rutina READST).
Registros afectados: A,X,Y.
Descripción Esta rutina se utiliza para cerrar un determinado archivo. En el acumulador se carga el número del archivo el cual debe ser cerrado.
Pasos a seguir:
1) Cargar en el acumulador el número de archivo a cerrar.
2) Llamar a esta rutina.
Ejemplo:
LDA #$04; cerramos el archivo nro. 4
jSR $FFC3
El número de error 240 está asociado con la interfase RS232.

ALMACENAMIENTO DE LOS PROGRAMAS

No todos los programadores saben cómo se almacena su software en memoria. Algunos creen que las líneas ingresadas son guardadas en la memoria de la misma forma en que fueron tipeadas. Es importante conocer cómo se efectúa este procedimiento y aquí se lo explicamos.

Son muchas las cosas que suceden cuando comenzamos a programar. La primera de ellas es cuando el sistema está a la espera de un comando o de una línea Basic.

Para este caso, se dice que el sistema está en el bucle o lazo principal.

Al comenzar a tipear una serie de caracteres (ya sea un comando o una línea), éstos se van almacenando en el buffer de entrada (desde la dirección $0200 hasta la $0258).

Al oprimir la tecla RETURN se chequea que la cantidad de caracteres no supere los 88. De otra manera se imprimirá el mensaje STRING TOO LONG.

Si, efectivamente, el número de caracteres es inferior a los 88, entra en juego la rutina que se encarga de traducir los caracteres tipeados a un código que sí entiende la computadora y que se conoce con el nombre de token.

Si el código ASCII del primer caracter corresponde a un número, se parte de la suposición que se ingresa una línea de programa. Si sólo se ingresó el número de línea, el intérprete comprueba si esa línea ya existe. En caso afirmativo, procede a borrarla de la memoria.

Caso contrario (es decir si se ingresa número de línea más texto) se codifica y se almacena.

Si, en cambio, ese código ASCII no corresponde a ningún número, se codifica y se bifurca a la rutina que ejecuta un comando Basic.

Así como cada caracter está codificado en ASCII (la "A" vale 65, la "B" vale 66, etc.) los comandos también están codificados en determinados valores (tokens).

Por ejemplo, el comando PRINT se codifica en 153 ($99 hexa), POKE en 151 ($97) y así se continúa.

¿Cuál o cuáles son las ventajas de trabajar con este método?

La primera es el ahorro de memoria. En vez de utilizar, por ejemplo, cuatro bytes para almacenar el comando POKE, sólo se utiliza un byte (su token).

La segunda es que se logra mayor velocidad en la ejecución de los programas. Imaginémonos el tiempo que se perdería en comparar con los cuatro caracteres de POKE (primero al P, luego la O, así hasta llegar a la E). A través de este método, sólo se compara con el token del comando.

Regresando al tema, otra de las cosas que debemos saber es cómo trabajan, o mejor dicho cuál es la estructura de las listas encadenadas.

Como su nombre lo indica, éstas se unen a través de un encadenamiento sucesivo.

Supongamos que cada elemento de la lista está formado por dos campos (campo A y campo B). El primero contiene la dirección o ubicación del próximo elemento de la lista. El segundo contiene datos y texto.

Para indicar fin de encadenamiento se almacena, en un determinado elemento, el valor de 0 en el campo A. El gráfico 1 representa una lista encadenada de 3 elementos.

Además, supondremos que existen determinados "casilleros" que indican cuál elemento es el primero de la lista y cuál es el último.

A estos casilleros los denominaremos C1 y C2.

En nuestro caso C1=1 y C2=3, lo que significa que el encadenamiento comienza con el primer elemento y culmina con el tercero.

A primera vista son dos las ventajas que surgen de trabajar con este tipo de estructuras.

Supongamos que debemos insertar el elemento 5 entre el 1 y el 2. Si hubiésemos trabajado con vectores, tendríamos que correr todos los elementos una posición hacia arriba para poder insertarlo en el vector.

Aquí, sólo debemos modificar los punteros del elemento 1. En vez de apuntar al elemento 2, hacemos que éste apunte al elemento 5 el cual hacemos que apunte al segundo elemento (gráfico 2).

Algo similar ocurre cuando borramos.

Analizando el esquema del gráfico 1, para borrar el elemento 2, hacemos que el elemento 1 apunte al 3 (gráfico 3).

En los equipos Drean Commodore (al igual que en los Commodore americanos) se utiliza este tipo de estructura.

Cada línea se almacena en memoria de la siguiente manera: los primeros dos bytes representan la dirección de la próxima línea Basic (primero byte bajo y luego byte alto).

El tercer y cuarto byte representan el número de la línea Basic (también en formato byte bajo-alto).

El quinto byte representa el token del comando.

Desde el sexto se almacenan los códigos ASCII de los caracteres que constituyen la línea.

El último byte es 00 representando, así, el fin de línea.

El fin de encadenamiento se indica, también, con 00.

Los que tienen la C-16 o la C-128, pueden utilizar el monitor residente para visualizar el almacenamiento de los programas.

Aquellos que tengan la C-64, deberán

recurrir a un monitor como, por ejemplo, el HESMON.
Analicemos a través del siguiente ejemplo, cómo es el formato hasta aquí descripto. Tipeen las siguientes líneas:

```
10 REM LINEA 10
20 REM LINEA 20
30 REM LINEA 30
```

Ingresen al modo monitor y tipeen
M1001 (usuarios de la C-16)
M4001 o M1C01 (usuarios de la C-128)
M0801 0820 (usuarios de la C-64)
Lo que verán será una serie de valores. Nosotros efectuaremos la explicación para uno de ellos. Es exactamente igual para cualquier tipo de Commodore; pero aquí analizamos como ejemplo la C-64.

```
0801 10 08 0A 00 8F 20 4C 49
0809 4E 45 41 20 31 30 00 1E
0811 08 14 00 8F 20 4C 49 4E
0819 45 41 20 32 30 2D 08 1E
0821 00 8F 20 4C 49 4E 45 41
0829 20 33 30 00 00 00 00 00
```

Los primeros dos bytes (contenidos en las direcciones $0801 y $0802) representan la dirección de la próxima línea del texto basic (en formato byte bajo-byte alto). En nuestro caso, ésta es $0810.

Los siguientes dos bytes, correspondientes a las direcciones $0803 $ 0804, representan el número de línea actual (número de línea $000A=10; también se almacenan en formato byte bajo-alto).

El quinto byte (dirección $0805) representa el token de la sentencia REM. ($8F).

Desde la dirección $ 0806 hasta la dirección $ 080E, se almacenan los códigos ASCII de LINEA 10.
El último byte (dirección $080F) representa fin de línea.
Continuamos con la segunda línea, la cual es exactamente igual (en formato).

Gráfico 2

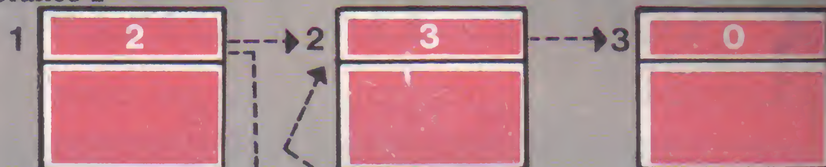


Gráfico 1

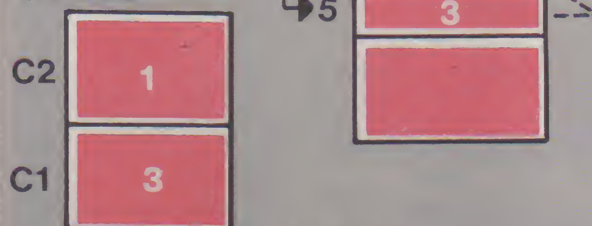
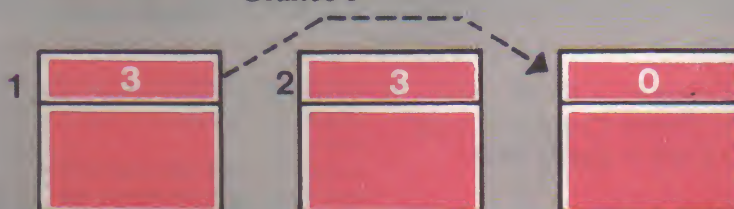


Gráfico 3



Las direcciones $0810 y $0811 contienen la dirección de la próxima línea Basic (es decir la número 20).
El encadenamiento continúa de esta manera hasta que el puntero de la próxima dirección de almacenamiento de la próxima línea señale a un lugar donde sea 00 (caso de las direcciones $081E y $081F, cuyos contenidos señalan a la dirección $082D, en donde se encuentra 00).
Les sugerimos que inserten una nueva línea y vean cómo se modifica este formato.
Igualmente pueden borrar una línea y ver cómo quedó en memoria el programa Basic.
Finalmente les dejamos la tabla 1 la cual representa los códigos token de los comandos comunes entre la C-16, C-64 y C-128.
Los usuarios de la C-16 y C-128 pueden descubrir los de los nuevos comandos, utilizando el monitor.

PRODE INTERNACIONAL

```
PRODE INTERNACIONAL

PRIMERAMENTE SE DEBERAN INGRESAR
LOS PARTIDOS A DISPUTARSE.
LUEGO, CADA PARTICIPANTE EFECTUARA EL
RESPECTIVO PRONOSTICO DE CADA UNO
DE ELLOS. CUANDO HALLA CULMINADO
EL MUNDIAL, LA COMPUTADORA INDICARA
QUIEN FUE LA PERSONA QUE ACERTO
MAS PARTIDOS.

OPRIMA UNA TECLA
```

Clase: Juego
Comp.: Drean Commodore 64
Conf.: C-64 + Drive 1541

Para que puedan intervenir en el Mundial de Fútbol con sus computadoras hogareñas, les presentamos este soft.
El objetivo del juego es que varios participantes (a lo sumo cinco) realicen sus respectivos pronósticos con respecto al resultado de cada uno de los partidos a efectuarse.
Luego, cuando éste haya finalizado, la computadora les dirá qué jugador acertó mayor cantidad de resultados.
Además, podrán utilizarlo para almacenar los eventos más importantes de la Copa del Mundo.

Comienza describiendo a grandes rasgos las condiciones y objetivos del juego.
Luego nos pedirá que ingresemos el total de partidos (no mayor que 70). De esta manera debemos ingresar cada uno de los encuentros.
Se nos informa que con un valor de 0 (cero) en respuesta al total de equipos, se entenderá que los partidos ya han sido almacenados.
Igualmente se nos pedirá la confirmación de esta orden.
Una vez que hayamos terminado esta tarea, debemos ingresar la cantidad de participantes.
A continuación se nos preguntará si ingresamos los nombres (opción 1) o si imprimiremos (opción 2).
La primera debe ser elegida cuando ingresamos por primera vez los nombres de los participantes.
La segunda se bifurcará a la sección del programa que decidirá quién fue el que obtuvo el mayor puntaje.
En caso de optar por la número 1, el programa comenzará a pedir los nombres de los participantes. Luego les pedirá, a cada uno, que formulen el correspondiente pronóstico de los partidos antes tipeados.
Es decir que imprimirá los pares de equipos y, además, preguntará si el resultado será local, empate o visitante.
Se entiende que un resultado local corresponderá al triunfo del equipo primero ingresado, mientras que un resultado visitante al segundo tipeado.
Luego de que cada participante haya realizado su "boleta"; se almacenarán los datos en disco.
A la hora de decidir sobre el ganador, se deberá responder 0 (cero) a la pregunta referente a la cantidad de archivos.
De esta manera el programa nos pedirá que ingresemos los resultados finales, para así poder comenzar a evaluar quién fue el ganador.
Respecto a cómo se tipeó el programa, continuamos con el método utilizado en "PROTECTOR DE ARCHIVOS".

Para aquellos que no dispongan de unidad de disco, pueden modificar las sentencias de OPEN de las líneas 114 y 510 poniendo un 1 en lugar de 8.

El único caracter especial no comentado es el correspondiente a ¡ctrl8! el cual significa que se debe oprimir la tecla CONTROL y 8 simultáneamente.

```
1 REM PRODE INTERNACIONAL
2 PRINT"!clr!":CR$=CHR$(13):DIMF,SC,MA
3 POKE53280,1:POKE53281,2:PRINT"!ctrl8!"
4 DIME1$(70),E2$(70),R$(5,70),RA$(70)
10 PRINT"!clr!!rvs on!!12es!PRODE INTERNACIONAL!9es!!rvs off!"
20 PRINT"!2cr ab!!es!PRIMERAMENTE SE DEBERAN INGRESAR"
22 PRINT"!es!LOS PARTIDOS A DISPUTARSE."
```

```
24 PRINT"!es!LUEGO, CADA PARTICIPANTE EFECTUARA EL"
26 PRINT"!es!RESPECTIVO PRONOSTICO DE CADA UNO"
28 PRINT"!es!DE ELLOS. CUANDO HALLA CULMINADO"
30 PRINT"!es!EL MUNDIAL, LA COMPUTADORA INDICARA"
32 PRINT"!es!QUIEN FUE LA PERSONA QUE ACERTO"
34 PRINT"!es!MAS PARTIDOS."
36 PRINT"!2cr ab!":GOSUB900
40 PRINT"!clr!!3cr ab!!es!A CONTINUACION INGRESE TODOS LOS"
42 PRINT"!es!PARTIDOS A EFECTUARSE."
44 PRINT"!2cr ab!":GOSUB900
46 PRINT"!clr!!rvs on!!12es!INGRESO DE PARTIDOS!9es!!rvs off!"
47 PRINT"!cr ar!!rvs on!!3es!0 INDICA QUE YA HAN SIDO
   INGRESADOS!2es!!rvs on!"
48 INPUT"!es!TOTAL DE PARTIDOS!2es!0!3cr iz!";P%
50 IFP%>70THENPRINT"!cr ar!!es!DEMASIADOS PARTIDOS":GOTO46
51 IFP%<>0THEN54
52 INPUT"!cr ab!!es!ESTA UD SEGURO (S/N)!2es!N!3cr iz!";DI$:
   IFDI$="N"THEN46
53 IFDI$="S"THEN64
54 FORI=1TOP%
55 FORI=1TOP%
56 INPUT"!es!EQUIPO1, EQUIPO2";E1$(I),E2$(I)
58 NEXTI
60 PRINT"!2cr ab!":GOSUB900
64 PRINT"!clr!!rvs on!!9es!INGRESO DE PARTICIPANTES!7es!
   !rvs off!"
66 INPUT"!2cr ab!!es!TOTAL DE PARTICIPANTES!2es!1!3cr iz!";PA%
```

```
68 IFPA%>5THENPRINT"!cr ab!!es!DEMASIADOS PARTICIPANTES":GOTO66
70 INPUT"!cr ab!!es!INGRESA (1) O IMPRIME (2)!2es!
   1!3cr iz!";OP%
71 IFOP%<>1ANDOP%<>2THEN64
72 IFOP%=2THEN500
74 FORI=1TOPA%:PRINT"!es!NOMBRE PARTICIPANTE ";I
76 INPUT"!es!";PA$(I):NEXT
78 PRINT"!2cr ab!":GOSUB900:PRINT"!clr!"
80 FORJ=1TOPA%
82 PRINT"!rvs on!TURNO DE ";PA$(J)
84 PRINT"DETERMINE POSIBLES RESULTADOS"
86 FORI=1TOP%
90 PRINTE1$(I);" VS ";E2$(I)
92 INPUT"LOCAL(L),EMPATE(E) O VISITANTE(V)!2es"L!3cr iz!";OP$
94 IFOP$="L"THENR$(J,I)="L":GOTO100
96 IFOP$="E"THENR$(J,I)="E":GOTO100
98 IFOP$="V"THENR$(J,I)="V":GOTO100
99 GOTO92
100 NEXTI:PRINT"!clr!":NEXTJ
110 PRINT"!2cr ab!":GOSUB900:PRINT"!clr!"
112 PRINT"!cr ab!!2es!GRABANDO ARCHIVO DE PARTICIPANTES"
114 OPEN2,8,2,"@0:PARTICIPA,S,W"
115 PRINT#2,P%,CR$,PA%
118 FORI=1TOPA%:FORJ=1TOP%
120 PRINT#2,PA$(I)+CR$+E1$(J)+CR$+E2$(J)+CR$+R$(I,J)+CR$
122 NEXTJ:NEXTI
124 CLOSE2
126 GOTO10
```

```
500 REM IMPRIME RESULTADOS
505 PRINT"!clr!!rvs on!!10es!IMPRIME RESULTADOS!12es!!rvs off!"
507 PRINT"!2cr ab!!es!CARGANDO ARCHIVO"
508 REM CARGAMOS ARCHIVO
510 OPEN2,8,2,"PARTICIPA,S,R"
512 INPUT#2,P%,PA%
513 FORI=1TOPA%
516 FORJ=1TOP%
520 INPUT#2,PA$(I),E1$(J),E2$(J),R$(I,J)
524 NEXTJ:NEXTI:CLOSE2:REM F=-1
528 PRINT"!clr!!rvs on!!7es!INGRESO DE RESULTADOS
    FINALES!4es!!rvs off!"
530 PRINT"!2cr ab!!es!INGRESE EL SCORE FINAL DE LOS PARTIDOS"
```

```
534 PRINT"!es!QUE SE IMPRIMIRAN A CONTINUACION"

540 PRINT"!es!TENIENDO EN CUENTA QUE LOS RESULTADOS"

544 PRINT"!es!PUEDEN SER 'L', 'E' O 'V'"

550 REM ** IMPRIME RESULTADOS STOR **

552 FORI=1TOP%

554 PRINTTAB(2);E1$(I);" VS ";E2$(I)

558 INPUTRA$(I)

560 IFRA$(I)<>"L"ANDRA$(I)<>"E"ANDRA$(I)<>"V"THEN554

562 NEXTI

570 PRINT"!2cr ab!":GOSUB900

575 PRINT"!clr!!rvs on!!3es!DETERMINA PUNTAJE
    DE PARTICIPANTES!3es!!rvs off!"

580 FORI=1TOPA%

585 FORJ=1TOP%

590 IFR$(I,J)=RA$(J)THENSC(I)=SC(I)+1

595 NEXTJ:NEXTI

600 PRINT"!2cr ab!!es!LOS PUNTAJES DE CADA PARTICIPANTE
    SON:":MA=SC(1):J=!

605 FORI=1TOPA%:IFSC(I)>MATHENMA=SC(I):J=I
610 PRINT" PARTICIPANTE ";PA$(I);" : ";SC(I)
615 NEXT
620 PRINT"!2cr ab!!es!GANADOR :";PA$(J)

625 PRINT"!es!CON UN TOTAL DE ";SC(J);" ACIERTOS"

627 FORI=1TOPA%:SC(I)=0:NEXT

630 PRINT"!2cr ab!":GOSUB900:GOTO10

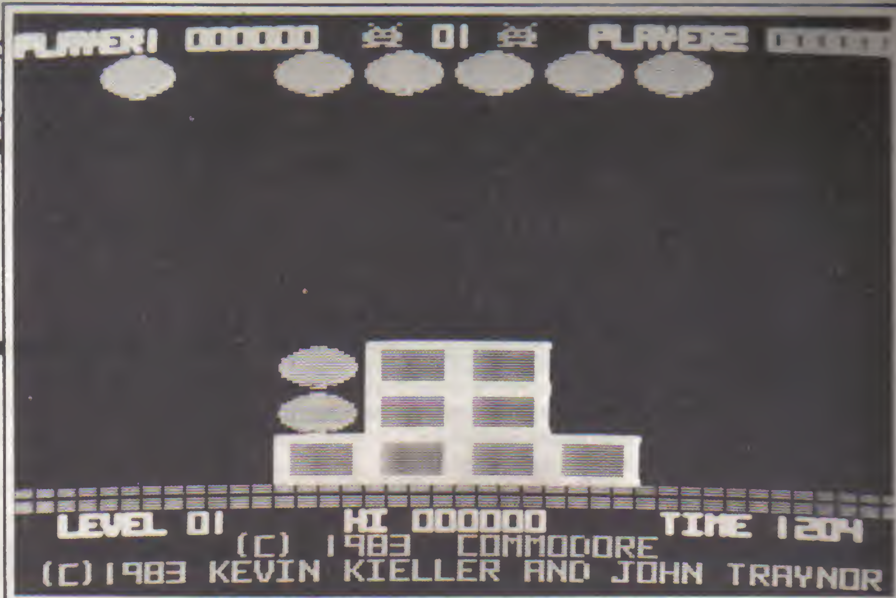
900 REM GETA$

902 PRINTTAB(11);"!rvs on!OPRIMA UNA TECLA!rvs off!"

904 GETA$:IFA$=""THEN904

906 RETURN
```

JACK ATTACK



Computadora: Drean Commodore 16
Rating Total: B⁻
Creatividad: B
Documentación: B
Profundidad del juego: B⁻
Valor en relación al costo: C
Mantiene el interés: C
Editor: Camelot
Desafío: Mediano.

JACK ATTACK nos hace acordar a aquella publicidad de jabón en polvo, en donde un "glóbulo" bueno se comía a los malos (la suciedad de la ropa).

Por supuesto que el juego no tiene nada que ver con aquélla y su fin no es la de vender jabón en polvo, sino que es el de pasar un rato más que divertido.

JACK ATTACK (Jack ataca) es un "globulito" que debe comer la mayor cantidad de raros monstruos, similares a globos.

Estos tratarán de matarlo, lo que se logrará cuando uno de esos bichitos se suba a nuestro Jack.

Además de ellos, se encuentran unos cubos que podemos mover para incrementar nuestro score. Eso sí: debemos tener cuidado de no mover los de abajo. De otra manera una pila de cubos caerá sobre nuestro amigo aplastándolo por completo.

Esta es otra de las cualidades que fomenta el juego: el ingenio.

Descubrir la mejor forma de trasladarlos no es tarea sencilla, sino lo creen, pregúntenle al Jack de la redacción!!!

El juego comienza al presionar el botón de disparo del joystick. De esta manera seleccionamos cuántos jugadores participarán (1 ó 2) y la dificultad del juego (de 1 a 9).

Esta tarea se realiza moviendo el joystick y posicionándose sobre la elegida.

El nivel inicial está formado por nueve cubos. Jack puede saltar hasta una altura igual a los tres cubos y, además, desplazarse hacia la izquierda o derecha.

Los globos enemigos tratarán de pisarnos, lo que debemos evitar escapando de ellos o tratando de aplastarlos.

Por cada globo aplastado, se nos sumará 100 puntos a nuestro score.

Otra forma de eliminarlos es aplastándolos con los mismos cubos.

El movimiento de los cubos se logra oprimiendo el botón de disparo.

Estos se pueden mover hacia la derecha o hacia la izquierda. Sólo se pueden desplazar los cubos que estén en el mismo nivel (de altura) que Jack.

Cada bloque desplazado incrementará en dos puntos a nuestro score.

Recuerden que Jack puede ser aplastado por los cubos cuando éste trate de moverlos.

El puntaje se incrementa, también, si terminamos un nivel, es decir destruir a todos los "globulitos", en el menor tiempo posible.

En este caso, la cantidad de tiempo restante es pasada a nuestro score.

Cuando hayamos superado los 20000 puntos, se integrará un nuevo Jack al equipo.

En ciertos niveles, donde el medio es agua, debemos tratar de ir formando especies de puentes con los cubos y, por supuesto, darles un buen pisotón a los "globulitos".

JACK ATTACK es un juego sumamente divertido, donde no sólo hay que eliminar "objetos" sino que se debe agudizar el ingenio para mover los bloques de la mejor forma posible.



A: SUPERIOR	D: REGULAR
B: MUY BUENO	E: MALO
C: BUENO	F: INACEPTABLE

1985 AFTER DAY

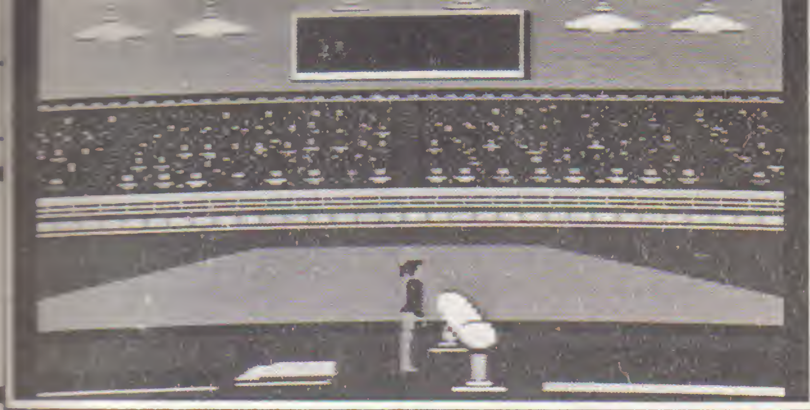
Computadora: Drean Commodore 64
Rating Total: C⁻
Creatividad: D
Documentación: C
Profundidad del juego: D
Desafío: Mediano
Valor en relación al costo: C
Mantiene el interés: C⁻
Editor: Peek

Luego de la serie de Moon Mission (misión lunar), donde se comenzaba a percibir la mejora de los juegos de video para la C-64, los productores de software comenzaron a buscar nuevas técnicas de programación. Estas debían mejorar el movimiento de los objetos logrando, así, aumentar la animación general del juego.
1985 AFTER DAY es uno de los primeros desarrollado con tales técnicas.
El objetivo del juego es comandar una nave espacial a través de varios túneles y laberintos.
El juego comienza cuando se terminan de cargar los tanques de combustible. Una vez hecho esto, la plataforma que mantiene la nave comenzará a deslizarse para que podamos despegar.
Luego de que se hayan soltado las amarras que sostienen nuestra nave, ésta comenzará a descender como consecuencia de la gravedad del planeta.
Para contrarrestar este efecto, debemos acelerar la nave moviendo hacia adelante la palanca del joystick.
En caso de darle demasiada aceleración, podemos chocar contra las paredes del túnel subterráneo. De todas maneras podemos girar la nave 180 grados y volver a acelerar. Así conseguiremos frenar la nave.
Luego de haberla estabilizado, debemos comenzar a movernos hacia la derecha o hacia la izquierda buscando la salida del túnel.
No hay que descuidar las paredes; de otra manera sufriremos un "pequeño" dolor de cabeza.
Antes de abandonar el túnel, debemos esperar un platillo volador, que, periódicamente, atraviesa toda la superficie del planeta. ¡En caso de estar en su camino, recordaremos aquel dolor de cabeza!
Si sobrevivimos a este platillo, tenemos que dirigirnos hacia una de las lunas del planeta.
Cada una de ellas posee un color distinto. Aquí no hay que preocuparse: no se puede chocar contra ese astro.
Una vez que nos hayamos acercado lo suficiente, la pantalla cambiará de escenario.
Este será más que hostil ya que, por un lado, se nos disparan misiles desde distintos sectores y, por el otro, el plato volador nos sigue buscando para sugerirnos que tomemos alguna aspirina.
Nuestro objetivo es aterrizar sobre puntos bien distinguidos (aquellos que no sean rampas de misiles).
Sobre el lugar de aterrizaje se encuentra una especie de meteorito que debemos destruir para poder aterrizar.
Esto se consigue oprimiendo el botón de disparo del joystick. Así desintegraremos la roca y podremos aterrizar.
El score se va incrementando de acuerdo con el número de aterrizajes obtenidos.
Cada vez que seamos destruidos (ya sea por el plato volador , el cual es similar a la de la serie OVNI; o por algún misil), el juego comenzará desde el abastecimiento de combustible.

SUMMER GAMES

Rating total: A⁻
Creatividad: B⁺
Documentación: B
Profundidad de juego: B
Desafío: difícil
Valor en relación al costo: B⁻
Mantiene el interés: B
Computadora: Drean Commodore 64
Editor: Peek

Summer Games (juegos de verano) nos hace recordar a las famosas olimpíadas que, cada cuatro años, se llevan a cabo en algún lugar del mundo. Esta es una representación exacta de uno de los eventos deportivos más importantes.
Comienza con la ceremonia inicial, en donde el atleta Jim Butefield transporta la antorcha olímpica, dando así inicio a los juegos.
Una bandada de palomas invaden el estadio, produciendo una gran emotividad entre el público presente.
Luego de finalizar el acto inaugural, debemos seleccionar en dónde competiremos. De esta manera podremos competir o practicar en uno o más juegos.
También se nos permite cambiar el número de joystick: podemos elegir sobre qué port lo conectaremos.
En este primer menú también observamos los records mundiales y, si queremos emocionarnos nuevamente, ver otra vez la ceremonia inicial.
Luego de seleccionar la competición, debemos ingresar nuestro nombre e indicar a qué país representaremos.
Esto se realiza posicionando el cursor sobre una de las banderas que constituyen la tercer pantalla.
Ellas son Francia, Italia, Australia, Canadá, Unión Soviética, Japón, Brasil, México, España, Irlanda, Inglaterra y USA.
Después de oprimir el botón del joystick, debemos ponernos de pie, ya que comenzará a interpretarse el himno nacional de ese país constituyendo, así, el segundo momento de gran emotividad.



El ingreso de datos finaliza cuando presionamos RETURN. A continuación se nos preguntará si los datos ingresados son correctos. De esta manera podemos corregirlos, en caso de tiperalos incorrectamente.
La próxima pantalla en aparecer es la que nos pide el evento en el cual competiremos. Estos pueden ser garrocha, saltos ornamentales, posta de 4x400, 100m, gimnasia, posta estilo libre, 100m estilo libre y tiro al plato.
A través del joystick seleccionamos el juego. Cuando oprimamos el botón, comenzará a cargarse la competencia deseada.
En tiro al plato debemos dispararle a los platos que se tiran al aire desde ambos extremos de la pantalla. Nuestra posición se va trasladando cada vez que disparamos, independientemente de acertar en el blanco.
Luego de que un jugador finalice sus disparos, continúa el turno del siguiente. La competencia culmina cuando hayan participado todos.
Luego se imprimen las medallas que cada uno obtuvo, en función del puntaje logrado. Se toca el himno del país que representa al ganador. Tercer momento de gran emotividad.
En saltos ornamentales, otra de las competencias, se deben realizar los mejores saltos. Por supuesto, esta pantalla está formada por la pileta, el trampolín, el jurado que califica nuestro salto y "el maravilloso público".
Primero debemos realizar saltos hacia adelante tratando de hacer varios mortales, los que se hacen a través del joystick.

Se nos calificará de acuerdo a cómo entremos en el agua. Se imprimen los puntajes otorgados por cada juez.
SUMMER GAME puede ser considerado como uno de los grandes dentro de los juegos de tercera generación.
Dispone de gran cantidad de gráficos animados con una excelente calidad en interpretación de melodías.

VIDUZZLES

Rating Total: B
Creatividad: A
Documentación: B
Gráficos: B
Computadora: Drean Commodore 16
Editor: Camelot

Seguramente la mayoría de nosotros hemos armado rompecabezas no sólo los días de lluvia.
Claro que, como dice la canción, el tiempo pasa y nos vamos poniendo viejos..., y dejamos de lado esta actividad.
Sin embargo, la industria informática sigue avanzando día a día, volcando a la computadora las más diversas aplicaciones.
Una de ellas es la de armar rompecabezas utilizándola.
VIDUZZLES es un juego cuyo objetivo consiste en armar un rompecabeza en el menor tiempo posible, partiendo de imágenes que suministra la computadora.
Estas son las de un payaso, un buho y un perro. Todas graficadas en alta resolución y con varios colores.
Primeramente debemos seleccionar las cantidad de fichas que tendrá el rompecabeza. Estas pueden ser 25 ó 50 piezas, lo que hará que el juego sea fácil o difícil.
Cada elección se realiza a través del joystick, posicionando un rectángulo rojo alrededor de la opción deseada.
A continuación debemos decidir sobre el contorno del rompecabezas.
Finalmente elegimos la imagen (perro, buho o payaso) y comenzamos a rompernos la cabeza.
La pantalla se divide en dos: en la primera mitad los perímetros de la imagen. En la otra mitad las fichas.
La posición del cursor se indica a través de una flecha, que representa los movimientos del joystick.
La pieza se elige posicionando la flecha sobre ella y, luego, oprimiendo el botón de disparo.
La ficha será resaltada para indicarnos que fue seleccionada.
Ahora debemos elegir la ubicación dentro de los contornos y oprimir nuevamente el botón de disparo.
Si la ubicación fue la correcta, la pieza ocupará la posición, desapareciendo del "pilón" de fichas.
Si no, seguirá incrementándose el tiempo del juego.
En caso de estar más que perdidos, podemos pedir HELP que, más que ayuda, ¡es un grito de socorro!
De esta manera se imprimirá nuevamente la imagen.
Otra de las opciones disponibles es la de WAIT (esperar) la cual suspende la ejecución de juego hasta que se oprima el botón de disparo.
A través de la opción STOP regresamos al menú principal.
Otra opción es DROP, la cual nos permite elegir la pieza que nos fue



quitada luego de pasar los tres segundos sin poder ubicarla en la posición correcta.
Cuando se logre ubicar todas las piezas, comienzan a sonar diversos sonidos indicándonos el logro de los objetivos.
Podemos reiniciar el juego tomando otra imagen.
VIDUZZLES es un juego muy entretenido y lo recomendamos para jugar no sólo en días de lluvia.

CORREO - CONSULTAS

Felicitaciones

Soy un asiduo lector de K64 y ahora de Drean Commodore. Pero en ninguna he visto alguna información sobre la Commodore 128 a pesar que ya se encuentra en nuestro mercado. En cuanto a la revista por fin algo exclusivo para los Commodormaníacos!

Oscar Sánchez - Cap. Fed.

Desde ya los felicito por la labor recién emprendida, que espero tengan tanto éxito como su otra publicación K64.
Acabo de comprarme la CBM64 y me gustaría que publiquen programas de juegos en lenguaje máquina, pues me interesa mucho todo lo referido al assembler.
Les deseo la mejor de las suertes ya que nos enseñan a usar provechosamente esta "caja de Pandora".

Alejandro Omar Eduardo Iglesias
Parque Chacabuco

Intercambios

Antes que nada deseo felicitarlos por la revista, es genial. Me gustaría a través de ella establecer correspondencia con jóvenes commodoreanos para intercambiar listados de programas, trucos e ideas. Gracias y hasta pronto.

Andrea Laura Nisoli
Comodoro Rivadavia 2263
(1872) Sarandí - Bs. As.

Me llamo Gerardo Pisani. Desearía intercambiar juegos de C-64 con otros usuarios.

Gerardo Pisani
Eugenio Ramírez 3232
TE.: 50-6256

Instrucción CLS

Hola. Tengo el agrado de escribirles en primer lugar para felicitarlos por la excelente revista que publican. Los programas son muy buenos y están bien impresos. Espero que sigan imprimiéndolos con letra grande.
Poseo una Drean Commodore 64 y quisiera hacerles unas preguntas:
1) La instrucción "CLS" no figura en el basic del Commodore. Qué instrucción la reemplaza?
2) Me podrían decir la dirección del club de usuarios de Drean Commodore.
Por último desearía que publiquen mis datos personales para intercambiar listados e información con otros usuarios de Drean Commodore.

Diego Hernán Brusasca
Hipólito Yrigoyen 7837
(1828) Banfield - Bs. As.

Entendemos que la instrucción CLS (Clear Screen) borra sólo la pantalla. En la C-64 ésto se lleva a cabo presionando SHIFT y HOME simultáneamente.
Con respecto a la dirección del club de usuarios, te la haremos llegar en la brevedad.

Continuamos con esta sección para que los lectores planteen sus consultas y sugerencias. Para eso deben escribir a Revista para usuarios de Drean Commodore, Paraná 720, 5to. Piso, (1017) Cap. Federal.

Periféricos

Les envío esta carta para comunicarles mis pedidos y sugerencias, las cuales me agradaría que tuvieran en cuenta.
Me llamo Gabriela Estévez, tengo 15 años. Hace poco tiempo adquirí una Drean Commodore 64. Me agrada mucho su revista; sus secciones son muy interesantes. Por mi falta de experiencia en este tema algunos artículos no los entiendo completamente pero pienso que más adelante los entenderé.
Poseo un libro ("La mejor programación por la práctica", Ediciones técnicas Rede) el cual estoy terminando de estudiar para comprender el lenguaje BASIC; estoy un poco confundida pues quiero comprar otro pero no sé cuál me beneficiará más.
Me gustaría que incluyeran una sección sobre bibliografía y, también, otra sobre los accesorios de esta computadora como ser la disketera, lápiz óptico, etc.
Les envío unos amables saludos.

Gabriela Elisa Estévez - Cap. Fed.

A partir del número 5 hemos comenzado a describir uno de los periféricos compatibles con la C-64.
Con respecto a la bibliografía, podemos decirte que es más que abundante la que se enfoca hacia los que recién se inician. Existen una gran cantidad de libros introductorios al BASIC. Te pedimos que visites alguna de las grandes librerías de la Capital Federal.

Diferencias

Me dirijo a ustedes para preguntarles qué diferencias hay entre el basic 3.5 de la C-16, el 2.0 de la C-64 y el 7.0 del C-128, así como también para preguntarles cómo hallar la memoria ocupada en la C-64.
Los felicito por la excelente publicación que hacen y querría saber cómo puedo hacer para conseguir los números 1 y 2.
Sin otro particular, los saludo muy atte.

Cristián Tissera - Córdoba

Hablar de las diferencias que hay entre la C-16, C-64 y C-128 nos llevaría prácticamente toda la revista. De todas maneras podemos realizar una pequeña descripción:
Trabajar con gráficos o con sonidos, en la C-64, resulta una tarea bastante complejo si sólo se dispone del basic 2.0. Ello demanda tener conocimientos sobre la ubicación del chip del video y, fundamentalmente, saber cómo trabaja. La C-128 y la C-16 disponen de comandos orientados al manejo de gráficos y de sonidos.
Para la programación estructurada, la C-16 y la C-128 también tienen sentencias orientadas, como ser REPET, WHILE y otras.
Si deseamos tener estos comandos, al igual que los anteriores, debemos tener un basic Plus como ser el Simon's Basic.
Otra de las diferencias importantes entre los tres equipos es su memoria libre. La C-16 tiene 12 Kb, la C-64 tiene 32 Kb y la de la C-128 es de 111 Kb.
Con respecto a tu pregunta sobre cómo hallar la memoria ocupada en la C-64, ésta comienza a partir de la dirección 2043 ($0801). A partir de aquí se encuentra nuestro programa en basic.
Los ejemplares atrasados los puedes conseguir en nuestra redacción cita en Paraná 720 5º Piso - (1017) Cap. Fed.

VICONEX LE SUMA UN NUEVO NEGOCIO A SU NEGOCIO

AHORA UD. PUEDE FINANCIAR A SUS CLIENTES CON NUESTRO EXCLUSIVO PLAN HASTA 12 MESES.

- Commodore 16
- Commodore 64
- Disketeras
  Drean Commodore 1541
- Impresoras
- Joysticks
- Accesorios
- Interface
- Programas de Juegos,
  Comerciales y Utilitarios

- Amplio surtido
- Stock permanente
- Los mejores precios

VICONEX
SU ALIADO EN COMPUTACION

ESMERALDA 870 - Capital Federal - Tel.: 312-3424
Av. de MAYO 767 - Capital Federal - Tel.: 34-8371/8357 30-3301
Av. de MAYO 702 - Ramos Mejía - Tel.: 658-3651

LA EMPRESA DE COMPUTACION QUE RESPALDA SU COMMODORE